



Universidad Nacional de la Patagonia San Juan Bosco

Tesina de Licenciatura en Informática

**Interfaz cerebro computadora (BCI):
Técnicas de Machine Learning aplicadas al
análisis de actividad neurológica mediante
un dispositivo de electroencefalografía
(EEG)**

Guillermo Eduardo De Jesus Jorge

Luis Miguel Luna

Tutor: Dr. Martin Bilbao

Mayo de 2022

Agradecimientos

Guillermo Eduardo De Jesus Jorge

Son muchas las personas que influyeron antes y durante el proceso de creación de este trabajo.

En primer lugar doy gracias a mi esposa **Doria** y **mi hija Emma** por entender y apoyar esta etapa de mi vida. Son y serán siempre mi inspiración, mi motivación y lo mas importante en esta vida.

Gracias a **mi familia**, mi viejo que me enseño que con constancia y humildad se puede, mi vieja por la vida, y mi hermana por su amor incondicional.

Gracias a **mis amigos** que son la familia que la vida puso en mi camino, por creer y brindarme su apoyo y entusiasmo en todo momento.

Gracias a mi amigo y compañero de Tesina **Luis Luna** por confiar, acompañar y compartir esta locura.

Agradecimiento especial al **Dr. Martín Bilbao**, que aceptó tutorear nuestro proyecto a pesar de su gran cantidad de obligaciones, brindando apoyo y orientación durante el camino, motivando siempre a mejorar.

Gracias a la universidad por sentar las bases necesarias para que nosotros estemos a esta instancia.

Gracias a mis médicos **Dr. Federico Cayol** y **Dra. Maria Laura Barrientos** que posibilitan que yo hoy esté acá y aunque no lo sepan de alguna manera incentivarón a enfocar mis esfuerzos para ayudar a otros.

Por último gracias **a mi** por no bajar lo brazos, por la constancia, por las ganas de seguir ante cualquier adversidad, por ganarle al cansancio, por ser inquieto, por mantenerme enfocado, por querer aprender siempre, por las ganas de ayudar y la auto motivación sabiendo que no era un camino fácil, pero se pudo, siempre soportado por todo el cariño y amor de mi gente.

Espero que este proyecto perdure en el tiempo y sea de utilidad para quien lo necesite, y agradezco a quienes destinen su tiempo para darle una mirada.

Around here, however, we don't look backwards for very long. We keep moving forward, opening up new doors and doing new things, because we're curious . . . and curiosity keeps leading us down new paths

Luis Miguel Luna

Antes que ha nadie debo agradecer y dedicar este proyecto de tesina a mi madre. Haber llegado hasta aquí, mi presente profesional y laboral, ha sido gracias su esfuerzo, su fortaleza y entereza ante la vida. Sin su ejemplo de bondad, sacrificio, ingenio y creatividad en el día a día no tendría las posibilidades que tengo hoy, ni hubiese tenido la motivación para continuar a pesar de los tropiezos que tuve durante la carrera.

También debo dedicar este proyecto de tesina a la memoria de mi padre, quién a pesar de haber dejado este mundo demasiado pronto, me ha acompañado siempre. Me dejó el ejemplo de un buen hombre trabajador, el recuerdo de su cariño y el deseo de que su hijo se convierta en profesional algún día.

Agradezco a Guillermo, un gran amigo y compañero, tanto en las últimas etapas de la carrera como durante los varios años que compartimos el ámbito laboral. Su impulso, positivismo y enorme capacidad me ayudaron a avanzar durante las ultimas cursadas, e hicieron de este proyecto una gran experiencia.

Muchas gracias a nuestro tutor, el Dr. Martín Bilbao, por acompañarnos en este proceso, por su guía, motivación y tiempo para consultas en horarios especiales por nuestras obligaciones laborales.

Muchas gracias de corazón a mis amigos más cercanos: Alejandro, Gonzalo, Juan Carlos y Pablo. Todos en distintos momentos me dieron su inestimable ayuda de una u otra manera, me motivaron a seguir con la carrera, a no bajar los brazos, me dieron el aliento que necesitaba y entendieron las ausencias, sobre todo durante este proyecto.

Finalmente, gracias a la universidad pública, que le da a tantas personas las oportunidades de mejorar su presente y futuro.

Resumen

Actualmente en el ámbito de la tecnología se ven constantemente innovaciones orientadas a facilitarle la vida a las personas. En los últimos tiempos uno de los campos que está teniendo mucho crecimiento a nivel mundial es el uso de Interfaces Cerebro Computadora (BCI), que permiten mediante la lectura de ondas cerebrales, tomar algún tipo de acción a partir de su análisis.

Las señales EEG son registros que se obtienen de acuerdo a la actividad eléctrica producida por el cerebro. Estas señales, luego de procesadas, pueden interpretarse como acciones producidas por un individuo.

Por otro lado, es innegable el enorme avance de las tecnologías de Inteligencia Artificial en la vida cotidiana de las personas. Uno de los campos de estudio de Inteligencia Artificial de mayor auge en los últimos tiempos es del Machine Learning, por lo que se cuenta con grandes fuentes de información de acceso público.

El propósito de esta tesina es desarrollar un prototipo de herramienta no invasiva, mediante el uso de la tecnología EEG y BCI, que permita caracterizar la actividad eléctrica del cerebro, procesarla y convertirla en información que pueda ser interpretada por una máquina y tomar acción en base a la misma.

Para lograr esto se utilizará un dispositivo de EEG comercial en conjunto con distintos modelos conocidos de Machine Learning. Con base en las métricas obtenidas, se seleccionará el modelo más apto para la herramienta propuesta. Cumplir este objetivo permitirá comprobar la viabilidad del uso de este conjunto de tecnologías para la mejora en la calidad de vida de las personas, y disponibilizar cualquier producto obtenido para el público en general.

Índice general

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos de la tesina	2
1.2.1. Objetivos específicos	2
1.3. Metodología propuesta	2
2. Estado del arte	3
2.1. Crecimiento y adopción	3
2.2. Resumen trabajo de Yongwook Chae, Sungho Jo and Jaeseung Jeong, Miembros de IEEE	5
2.2.1. Resultados obtenidos del trabajo de Yongwook Chae, Sungho Jo, Miembro, IEEE and Jaeseung Jeong, Miembro, IEEE	7
2.2.2. Conclusión del trabajo de Yongwook Chae, Sungho Jo, Miembro, IEEE and Jaeseung Jeong, Miembro, IEEE	7
2.3. Resumen trabajo de Luis Vergara, Yongrui Huang, Jianhao Yang, Pengkai Liao, and Jiahui Pan	7
2.3.1. Modelo de trabajo	8
2.3.2. Conclusión trabajo de Luis Vergara, Yongrui Huang, Jianhao Yang, Pengkai Liao, and Jiahui Pan	10
2.4. Resumen	10
3. Tecnologías Open Source	11
3.1. Introducción	11
3.2. ¿Qué son las tecnologías Open Source?	11
3.3. Beneficios del software Open Source	11

3.4. ¿Por qué utilizar tecnologías Open Source?	12
3.5. Resumen	13
4. Software utilizado	14
4.1. Introducción	14
4.2. JavaScript	14
4.2.1. Impacto	14
4.2.2. Entorno	15
4.2.3. Rendimiento	15
4.2.4. Compatibilidad Cross-Browser	16
4.2.5. Ejecución mono hilada	16
4.3. Node.js	16
4.3.1. Características	16
4.4. Firestore	17
4.5. TensorFlow.js	17
4.5.1. Arquitectura de TensorFlow.js	18
4.5.2. Integración en el ecosistema de TensorFlow	19
4.6. Vue.js	19
4.6.1. Estructura de componentes	19
4.6.2. Programación reactiva	20
4.6.3. Accesibilidad	20
4.7. Resumen	21
5. Hardware utilizado	22
5.1. Introducción	22
5.2. Mindwave MW001	23
5.2.1. Composición	23
5.2.2. Funcionamiento	23
5.2.3. Comunicación	24

5.2.4. Protocolo	24
5.3. Hardware para pruebas	25
5.4. Resumen	25
6. Inteligencia Artificial y Machine Learning	26
6.1. Introducción	26
6.1.1. Aplicaciones actuales	27
6.2. Machine Learning	28
6.2.1. Aprendizaje	28
6.2.2. Regresión lineal	31
6.2.3. Gradiente descendente	33
6.2.4. Naive Bayes	34
6.2.5. Probabilidad condicional	34
6.2.6. Teorema de Bayes	35
6.2.7. Árboles de decisión	37
6.3. Resumen	39
7. Arquitectura propuesta	40
7.1. Introducción	40
7.2. Arquitectura de solución propuesta	40
7.3. Soporte a la investigación	41
7.3.1. Conector Node.js para lectura de datos reales	41
7.3.2. Gráfico en vivo	47
7.3.3. Laberinto	48
7.4. Preparación del set de datos	49
7.4.1. Transformada rápida de Fourier (FFT)	50
7.5. Modelos de Machine Learning	50
7.6. Como entender resultados	51
7.7. Resumen	52

8. Resultados y discusión	53
8.1. Introducción	53
8.2. Obtención de datos	53
8.3. Datos primarios	54
8.3.1. Calidad datos primarios	54
8.3.2. Datos erróneos	54
8.4. Balanceo de clases	54
8.5. Pre procesamiento de datos	55
8.5.1. Normalización	55
8.5.2. Filtro de muestras no representativas	56
8.5.3. Transformada rápida de fourier (FFT)	56
8.5.4. Data augmentation: SMOTE	57
8.5.5. Matriz de desvío respecto a media y desvío estándar	57
8.5.6. Truncado de valores	58
8.5.7. Shuffle	58
8.6. Segmentación del set de datos	58
8.7. Modelos de clasificación	59
8.7.1. Regresión lineal simple	61
8.7.2. Naive Bayes	64
8.7.3. Árbol de decisión	67
8.8. Pruebas de modelo en tiempo real	71
8.8.1. Procesamiento de nuevos valores	71
8.8.2. Uso del casco	71
8.9. Resumen	73
9. Conclusiones y trabajos futuros	74
9.1. Conclusiones	74
9.2. Trabajos futuros	75

A. Guías de uso	76
A.1. Introducción	76
A.2. Prerequisitos	76
A.2.1. General	76
A.2.2. Software Thinkgear Connector	76
A.2.3. Node.js	77
A.3. Conector Node.js para lectura de datos reales	79
A.3.1. Requerimientos	79
A.3.2. Uso de API	79
A.3.3. Métodos de la API	80
A.3.4. Consideraciones en caso de problemas de conexión del casco .	81
A.4. Gráfico en vivo	82
A.4.1. Inicio de la aplicación	82
A.4.2. Uso de la aplicación	83
A.5. Laberinto	83
A.5.1. Inicio de la aplicación	84
A.5.2. Uso de la aplicación	84
A.5.3. Instrucciones de juego	85

Índice de figuras

2.1.	Número acumulado de Publicaciones	4
2.2.	Posiciones de los electrodos de EEG con respecto al sistema internacional 10-20.	6
2.3.	Adquisición y procesamiento de datos.	8
2.4.	Arquitectura del sistema de clasificación de expresiones faciales básicas	8
4.1.	Arquitectura de Node.js	16
4.2.	Arquitectura de TensorFlow.js, extraído y traducido a partir de [7]. .	18
4.3.	Estructura de árbol de componentes de Vue.js, extraído de [11]. . .	20
4.4.	Reactividad y ciclo de actualización de componentes, extraído de [11].	20
5.1.	MW001	23
5.2.	Adaptación de protocolo a solución propuesta	25
6.1.	Ejemplo de regresión lineal, función lineal y función de pérdida, sobre un set de entrenamiento de n puntos en el plano x, y , cada punto representando relación entre tamaño en pies cuadrados y precio de un conjunto de casas en venta.	31
6.2.	Flujo de trabajo bayesiano, imagen traducida por equipo, extraída de [16].	37
6.3.	Árbol de decisión del juego 'Piedra, papel o tijera' [17].	38
7.1.	Diagrama de interacción entre las distintas partes de la solución . .	41
7.2.	Estructura de colección	46
7.3.	Gráfico en vivo	48
7.4.	Juego de Laberinto	49

8.1.	Flujo de procesamiento de datos	55
8.2.	Etapa de pre procesamiento de datos	55
8.3.	Espectro de señal cruda.	56
8.4.	Espectro de señal sin procesar, y técnica de transformada rápida de fourier aplicada. Imagen tomada del libro [22]	57
8.5.	Set de datos original, 4 clases, 1000 valores por clase. Se de datos balanceado posterior a re-mapear las clases, 5 clases, 4000 valores por clase.	59
8.6.	Comparación de precisión obtenida al generar los modelos evaluados en base a los distintos criterios de filtros utilizados.	69
8.7.	Evolución de precisión de árbol de decisión en función de la cantidad de muestras.	70
A.1.	Contenido de instalación de ThinkGear Connector	77
A.2.	Instalador de Node.js	78
A.3.	Comando de instalación de aplicación Node.js	78
A.4.	Comando de ejecución de Conector para consulta de datos de lectura	80
A.5.	Dongle USB de conexión de casco “MindWave MW001”	80
A.6.	Dongle USB y casco EEG conectados de forma correcta	80
A.7.	Aplicación de visualización de gráficos en vivo.	82
A.8.	Comando de ejecución de aplicación web de Gráfico en vivo	82
A.9.	Aplicación web de Gráfico en vivo iniciada correctamente	83
A.10.	Barra de herramientas de aplicación de gráficos en vivo.	83
A.11.	Comando de ejecución de aplicación web de Laberinto	84
A.12.	Aplicación web de Laberinto iniciada correctamente	84
A.13.	Checkbox de lectura en vivo de Laberinto	84
A.14.	Caja de texto para URL de servidor de Conector Node.js para lectura de datos	85
A.15.	aplicación web de laberinto en ejecución	85

Índice de cuadros

2.1. Matriz de Reglas (R_i), Antecedentes (P) y Resultados (Q) para toma de decisiones	10
7.1. Notación para matriz de confusión	51
8.1. Matriz de confusión a partir de 8.6(a) precisión obtenida 14.03 % . . .	62
8.2. Matriz de confusión a partir de 8.6(b) precisión obtenida 27.28 % . . .	62
8.3. Matriz de confusión a partir de 8.6(c) precisión obtenida 18.41 % . . .	62
8.4. Matriz de confusión a partir de 8.6(d) precisión obtenida 17.55 % . . .	62
8.5. Matriz de confusión a partir de 8.6(e) precisión obtenida 21.22 % . . .	62
8.6. Matriz de confusión a partir de 8.6(f) precisión obtenida 42.65 % . . .	65
8.7. Matriz de confusión a partir de 8.6(g) precisión obtenida 36.38 % . . .	65
8.8. Matriz de confusión a partir de 8.6(h) precisión obtenida 44.5 % . . .	65
8.9. Matriz de confusión a partir de 8.6(i) precisión obtenida 36.55 % . . .	65
8.10. Matriz de confusión a partir de 8.6(j) precisión obtenida 38.6 % . . .	65
8.11. Matriz de confusión a partir de 8.6(k) precisión obtenida 41.07 % . . .	68
8.12. Matriz de confusión a partir de 8.6(l) precisión obtenida 38.28 % . . .	68
8.13. Matriz de confusión a partir de 8.6(m) precisión obtenida 48.33 % . .	68
8.14. Matriz de confusión a partir de 8.6(n) precisión obtenida 35.67 % . . .	68
8.15. Matriz de confusión a partir de 8.6(ñ) precisión obtenida 45 %	68

Lista de códigos fuente

7.1. Ejemplo de Paquete ThinkGear, tomado de [18]	43
7.2. Ejemplo de Paquete ThinkGear	44

Capítulo 1

Introducción

Las interfaces cerebro/computadora (BCI) permiten el control de hardware mediante el uso exclusivo de las ondas cerebrales. Una forma que está tomando cierto grado de relevancia en los últimos tiempos, es mediante dispositivos de electroencefalografía (EEG), que permiten adquirir, procesar y posteriormente analizar las ondas cerebrales de manera digital y en tiempo real.

Prototipar aplicaciones que empleen tecnología actual como Inteligencia Artificial (IA) en conjunto con EEG es un gran avance en la industria que puede resultar en una mejora en la calidad de estudios que se realizan actualmente, o incluso en la calidad de vida de las personas con movilidad reducida.

Las tecnologías BCI en conjunto con la Inteligencia Artificial conforman una tecnología emergente, de rápido crecimiento, que proporciona un medio de comunicación entre el cerebro (que “habla” silenciosamente) y dispositivos de biomonitorización. Éstas interfaces pueden verse como una colaboración entre un dispositivo, que permite el paso de señales eléctricas desde las neuronas, con un sistema externo: una computadora, un brazo robótico, u otro tipo de actuador que ejecuta una acción específica. Es una tecnología muy poderosa que no depende de la participación de ningún músculo o vías neuronales para completar la comunicación, el comando y por tanto la acción que se desea producir. [1] [2]

1.1. Motivación

Diversos impedimentos o limitaciones físicas del ser humano pueden ocasionar la dependencia de dispositivos o herramientas complementarias. Sin embargo, estos dispositivos o herramientas no siempre logran suplir totalmente las necesidades de la persona. En muchos casos las personas no son capaces de utilizar un periférico convencional tales como teclado, mouse, o una pantalla táctil, o mantienen dependencia de una silla de ruedas para su movilización, sin tener, sin embargo, la capacidad de manejarla por cuenta propia.

Se pretende investigar la viabilidad del uso de la tecnología EEG y BCI para desarrollar herramientas que faciliten y mejoren la calidad de vida de las personas

que viven en estas condiciones.

1.2. Objetivos de la tesina

Implementar un prototipo de herramienta no invasiva con la cual caracterizar la actividad eléctrica del cerebro, para luego procesarla y convertirla en un formato apto para visualización y análisis.

1.2.1. Objetivos específicos

- Analizar el estado del arte de la tecnología de dispositivos EEG - BCI.
- Crear una base de conocimientos asociada a lecturas de actividad cerebral.
- Estudiar modelos de predicción de ondas cerebrales mediante el uso de Machine Learning.
- Prototipar un modelo en base a lo estudiado.
- Documentar la experiencia en el uso de BCI.
- Disponibilizar el prototipo y los datos obtenidos para acceso público de manera gratuita.

1.3. Metodología propuesta

La metodología de realización de la tesina consta de un trabajo de investigación sobre el funcionamiento de BCI mediante dispositivos EEG.

Para la captura de actividad cerebral se empleará un dispositivo de lectura EEG que en primera instancia deberemos compatibilizar para uso en plataformas web. Logrado esto se disponibilizarán los set de datos obtenidos.

El proyecto de tesina dará lugar a la comprensión del uso de ondas cerebrales en conjunto a tecnología web y machine learning para la mejora de la calidad de vida de las personas con movilidad reducida.

Capítulo 2

Estado del arte

En la actualidad, la tecnología está presente y afecta a todos los aspectos posibles a la sociedad, desde la invención de la rueda hasta la inteligencia artificial. La necesidad de entender el cerebro humano motivo a investigadores durante las últimas décadas a desarrollar interfaces BCI eficientes, robustas y fiables. El funcionamiento básico siempre ha sido mediante sensores, procesando las señales adquiridas para obtener características de interés, y en última instancia la interacción con el entorno en base a la necesidad del usuario.

Desde un punto de vista clínico, los sistemas BCI basados en EEG han recibido un creciente interés debido a su naturaleza no invasiva, de nulo o bajo riesgo en comparación con otros sistemas invasivos y su forma fácil de guardar registros de los distintos eventos que se producen.

A continuación se presentará evidencia del crecimiento y adopción de tecnologías BCI basadas en EEG, comparación con otras tecnologías relacionadas y algunos trabajos recientes enfocados en dichas tecnologías.

2.1. Crecimiento y adopción

En años recientes la cantidad de artículos de índole científico con referencia a BCI se ha duplicado, debido al compromiso de una mayor comunidad en este campo, y la importancia real de la tecnología. En la imagen 2.1 se puede ver representado este crecimiento.

El mercado global de las tecnologías BCI se valoró en \$1.36 mil millones en 2019, y se prevé que alcance los \$3.85 mil millones para 2027, creciendo a una tasa compuesta anual del 14.3 % de 2020 a 2027.

Sobre la base del tipo, el segmento no invasivo dominó el tamaño general del mercado de tecnologías BCI en 2019, y se espera que mantenga el dominio, en tanto no surjan otras tecnologías que ofrezcan un mayor nivel de respuesta. Esto se atribuye a la ventaja de no requerir la inserción de un dispositivo extraño en el cerebro, lo que impulsa principalmente su crecimiento en el mercado.



Figura 2.1: Número acumulado de Publicaciones, que se refieren a BCI por tipo indexadas por IEEE Xplore, Web of Science, PubMed y Scopus. Gráfico de producción propia realizado con librería Echarts.

Para analizar las actividades cerebrales se utilizan cuatro paradigmas típicos en la señal de EEG, a saber, el potencial relacionado con el evento P300 (ERP), el potencial cortical lento (SCP), los ritmos sensorio motores (SMR) y el potencial evocado visual en estado estable (SSVEP).

Según informes del FDA, 1 de cada 4 adultos sufre trastornos cerebrales cada año, y aproximadamente 6 % de la población mundial sufre discapacidades graves. El aumento global de los trastornos cerebrales facilita los avances tecnológicos rápidos en pos de mitigar esta problemática mundial. Por ejemplo, en 2015, Mind Solutions, Inc. lanzó el dispositivo BCI más pequeño del mundo, “Synapse”. Estas innovaciones mejoran la eficiencia de las tecnologías BCI existentes y ayudan a los pacientes paralizados a controlar prótesis mediante procesos basados en el pensamiento, e incluso permite a pacientes con problemas de habla, mostrar sus pensamientos en una computadora.

Se espera que la creciente infraestructura de atención médica en economías en desarrollo, como India, brinde enormes oportunidades para el crecimiento de las tecnologías BCI en sus respectivas regiones. A medida que mejora la infraestructura sanitaria, se fomentan sistemas innovadores que mejoran la vida de las personas con discapacidad.

Según un informe de la OMS, alrededor de 82 millones de personas se verán afectadas por la demencia para el año 2030, lo que dirigirá la demanda potencial de BCI en los próximos años. Se prevé que la creciente aparición de afecciones neurodegenerativas como el Alzheimer, la enfermedad de Parkinson y la epilepsia impulsen la demanda de tecnologías BCI durante su período de pronóstico.

2.2. Resumen trabajo de Yongwook Chae, Sungho Jo and Jaeseung Jeong, Miembros de IEEE

El estudio [3] propone un novedoso sistema de navegación de robot humanoide accionado por el cerebro que permite el control directo para que los usuarios puedan seleccionar en un sistema basado en menús, primitivas de movimiento de bajo nivel (Por ejemplo, Parar, caminar hacia adelante, girar el cuerpo, girar la cabeza a la izquierda y giro de cabeza a la derecha) en lugar de primitivas de movimiento de alto nivel (por ejemplo, ir al lugar de destino limitado).

Para implementar un sistema de control directo, se utilizó un sistema BCI basado en potencia de banda para extraer los comandos de bajo nivel de la intención del usuario (por ejemplo, mano izquierda, mano derecha y pie).

Adicionalmente el sistema implementa un paradigma de flujo asincrónico, que al no depender del flujo de secuencial de señales, detecta continuamente no solo los estados de control intencional de usuario, sino también estado de NC (No control = actividad del cerebro en estado inactivo), habilitando al usuario a regular la sincronización del control y muestra un ITR (information transfer rate) mas alto que los sistemas netamente sincrónicos.

El paradigma de control fue diseñado para permitir al usuario seleccionar cinco movimientos diferentes, como girar la cabeza hacia la izquierda o hacia la derecha, caminar hacia la izquierda o hacia la derecha, o caminar en línea recta, usando solo tres comandos de motor basados en los estados dependientes posturales. Bajo este enfoque, se crea exitosamente la motricidad del robot humanoide.

El sistema consta de 3 subsistemas, el sistema BCI, sistema de interfaz y sistema de control, y se divide en 3 procedimientos principales, el entrenamiento offline, feedback online y control en tiempo real procesando 3 tipos de datos diferentes, la información visual procesada, las señales EEG medidas, y los comandos pre mapeados de movimiento.

En la etapa de entrenamiento offline consistió en un ciclo repetitivo de al menos 3 sesiones diarias, donde le solicitaron a los voluntarios imaginar movimientos de manos, pies y “descanso” estando en estado de reposo, y siguiendo indicaciones de una pantalla. Para evitar desvíos debido a respuestas en base a conocimientos previos, las imágenes eran aleatorias. Luego de cada sesión de entrenamiento offline, el sistema BCI se encarga de analizar los datos colectados y extraer solo características apropiadas, luego seleccionó los componentes de características informativas para finalizar entrenando clasificadores jerárquicos basados en componentes de las características seleccionadas.

Por otro lado, en la etapa de feedback online, el sistema muestra una interfaz visual donde permite ver una señal de destino y el estado mental clasificado usando las reglas de clasificación. Estos 2 procedimientos se repitieron hasta que la proporción de aciertos de la ejecución online fue superior al 75 %.

Durante el experimento, las señales EEG fueron obtenidas y grabadas a una frecuencia de 250HZ mediante 21 electrodos (F3, Fz, F4, FT7, FC3, FCz, FC4,

FT8, T7, C3, Cz, C4, T8, TP7, CP3, CPz, CP4, TP8, P3, Pz). Las señales EEG muestreadas de 9 canales (FC3, FCz, FC4, C3, Cz, C4, P3, Pz y P4) .

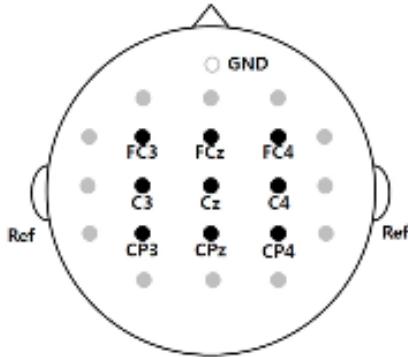


Figura 2.2: Posiciones de los electrodos de EEG con respecto al sistema internacional 10-20.

En la figura 2.2 las posiciones de los electrodos marcadas con círculos grises solo se utilizan en el cálculo del filtro espacial. Los nueve círculos negros indican las posiciones de los electrodos que se utilizan como canales de características principales. Todos los electrodos están referenciados a la mastoides izquierda y derecha.

Para enfatizar las imágenes obtenidas se aplicó un filtro laplaciano grande cuya formula tiene la siguiente forma.

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \quad (2.1)$$

Para traducir los datos de EEG provistos en sus comandos a movimiento apropiados para el robot humanoide, emplearon jerárquicamente el Clasificador de Actividad Intencional (IAC, por sus siglas en inglés) y el Clasificador de Dirección de Movimiento (MDC, por sus siglas en inglés). El IAC clasifica entre los estados NC y MI (motor imaginary, tareas que implementan imágenes estandarizadas para traducir ondas cerebrales de mano derecha, izquierda, pies, etc.). Si las señales fueron interpretadas como el estado MI por el IAC, entonces el MDC clasificó el estado MI específico como estados de “mano izquierda”, “mano derecha” o “pie”, basados en esta función, el IAC consiste las características de amplitud informativa que se extrajo de 2 de las bandas mas discriminantes durante el periodo, en tanto el LDA (análisis linear discriminante) es utilizados para entrenar el set de características para el IAC. Como resultado del entrenamiento los resultados negativos aportan NC classes, en tanto de los valores positivos provistos por la IAC obtienen MI classes. En base a la clase de resultado obtenido y mediante la función cuadrática de Fisher el MDC lo traduce a una clase de mayor nivel “left-hand”, “right-hand”, “foot”.

$$Fr^2 = \frac{\delta^2 between}{\delta^2 within} = \frac{(\mu_{rest} - \mu_{MI})^2}{\delta^2_{rest} + \delta^2_{MI}} \quad (2.2)$$

Para lograr mejor precisión y un último control antes de darle instrucciones al robot humanoide implementaron un sistema de feedback que permite evitar clasificaciones falsas buscando que cada comando aceptado se mantenga al menos por N

= 4 segundos, para informar al usuario la selección, el sistema indica mediante un cartel el porcentaje de coincidencia y una flecha de dirección que el humanoide debe tomar.

2.2.1. Resultados obtenidos del trabajo de Yongwook Chae, Sungho Jo, Miembro, IEEE and Jaeseung Jeong, Miembro, IEEE

Luego de todo este manejo de eventos, preparación del sistema, pruebas de comandos y reiterativas sesiones de entrenamiento lograron obtener una precisión entre 84.5 % y 87.3 %.

La performance del humanoide medida en base a el tiempo que requiere realizar una tarea, los metros que el robot debió caminar para cumplir una tarea, los puntos preseteados de navegación y las colisiones determinó que en promedio, para cada una de estas medidas el tiempo de control del robot fue 0.31 % mas lento que el control manual, y a excepción de 1 (un) individuo, la cantidad de pasos realizados por el robot siempre fue mayor bajo control del sistema BCI en comparación al control manual, logrando menos cantidad de puntos preseteados y mayor promedio de colisiones.

2.2.2. Conclusión del trabajo de Yongwook Chae, Sungho Jo, Miembro, IEEE and Jaeseung Jeong, Miembro, IEEE

Debido al número de colisiones y puntos de paso muestreados, se requerirá un sistema BCI más preciso o un sistema robótico mas complejo para evitar colisiones al controlar el robot humanoide en el mundo real. Además, para representar la función más compleja del robot humanoide además de la navegación, se requieren estados mentales adicionales.

2.3. Resumen trabajo de Luis Vergara, Yongrui Huang, Jianhao Yang, Pengkai Liao, and Jiahui Pan

Este artículo [4] propone dos métodos de fusión multimodal entre el cerebro y las señales periféricas para el reconocimiento de emociones. Las señales de entrada son EEG y expresión facial. Los estímulos se basan en un subconjunto de clips de películas que corresponden a cuatro áreas específicas del espacio emocional de excitación que maneja el encéfalo (felicidad, neutralidad, tristeza y miedo). Para detección de expresiones faciales implementaron un clasificador basado en redes neuronales que detecta los 4 estados emocionales básicos de una persona (felicidad, tristeza, miedo y neutral). Para adquisición de las señales EEG utilizaron un dispositivo Mindwave Mobile (Neurosky Inc., Abbotsford, Australia). La toma de decisión la basaron en la fusión de los 2 métodos buscando compensar los defectos de tener una sola fuente de información.

2.3.1. Modelo de trabajo

Utilizando el dispositivo Mindwave Mobile para adquisición y manejo de señales EEG y una cámara Logitech de 25 FPS a una resolución de 800x600 para la captura de expresiones faciales. Ambas fuentes de información fueron procesadas por separado 2.3. De acuerdo con el sistema estándar 10-20, las señales de EEG tomadas de la mastoides derecha utilizadas para el análisis se registraron a partir del electrodo “Fz”. Las impedancias de todos los electrodos se mantuvieron por debajo de $5\text{k}\Omega$.

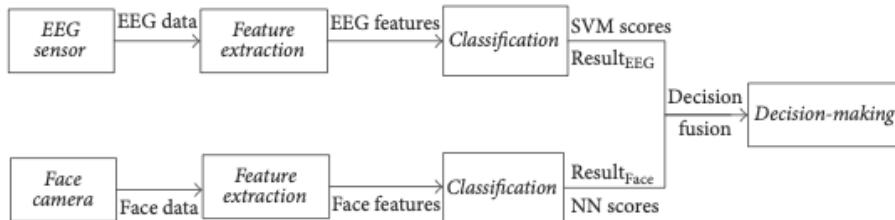


Figura 2.3: Adquisición y procesamiento de datos.

En tanto a la detección de expresiones faciales, se basaron en el sistema de Tiempo real AdaBoost cuyo algoritmo se basa en HAAR eigenvalue ¹. El resultado del clasificador es si esta imagen es un rostro humano, al encontrarlo, varian el tamaño de la imagen a 48px por 48px, luego implementan PCA (Principal Component Analysis) para reducir la dimensionalidad del dataset a un vector de 169 posiciones. Este vector es utilizado por una red neuronal prealimentada (feedforward neural network) de la cual obtienen 4 resultados; por último se normalizan mediante la función argmax en un rango de [0,1]. Estos 4 resultados normalizados son las salidas de la red pre-alimentada, son utilizados para representar los 4 estados emocionales. A continuación, en la figura 2.4, podemos ver un representación de esta red neuronal.

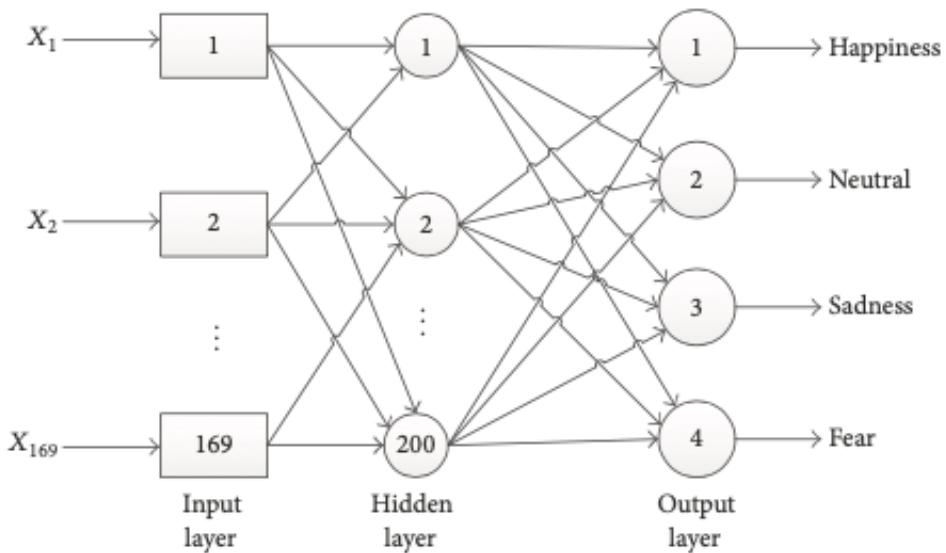


Figura 2.4: Arquitectura del sistema de clasificación de expresiones faciales básicas

¹Función encargada de detectar cambios de estado del valor propio de una variable, entre dos tiempos distintos.

La detección basada en EEG incluye dos etapas progresivas: extracción de características basada en PSD (Power Spectral Density) y clasificación mediante SVM (Support Vector Machines). Este proceso se encuentra diagramado en la figura 2.3.

Los datos del EEG se filtran en ocho bandas de frecuencia: delta (1-3 Hz); theta (4-7 Hz); alpha1 (8-10 Hz); alpha2 (11-13 Hz); beta1 (14-20 Hz); beta2 (21-30 Hz); gamma1 (31-40 Hz); y gamma2 (41-50 Hz).

Las características de PSD se calculan utilizando la Transformada de Fourier de Corto Tiempo (STFT) con una ventana de 1 segundo sin una ventana de Hanning superpuesta. En cuanto a la clasificación utilizaron dos clasificadores SVM lineales, uno para la clasificación de estados emocionales y otro para la clasificación de emociones e intensidades obteniendo las muestras de clasificación de la siguiente manera.

$$X_i = [\delta, \theta, \alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2]$$

$$Y_i = [1:\text{felicidad}, 2:\text{neutral}, 3:\text{tristeza}, 4:\text{miedo}]$$

$$Y_i' = [1:\text{débil}, 2:\text{moderado}, 3:\text{fuerte}]$$

Aplicando el primer clasificador de SVM entrenando a los vectores de características (X_i, Y_i) , obtienen cuatro puntajes (los valores de la función objetivo de SVM), denotados como S_2 ($J = 1, \dots, 4$). Donde J representa los cuatro estados emocionales. Luego entran el clasificador SVM con los vectores correspondientes a los 3 niveles de intensidad de emoción.

Con ambas salidas normalizadas entran en un modelo de clasificación de fusión decisión/nivel. El primero de estos representado como la suma de los resultados del clasificador SVM para las decisiones obtenidas.

$$\text{sum}_j = S1_j + S2_j \quad (J = 1, 2, 3, 4) \quad (2.3a)$$

$$r_s = \arg_j \max(\text{sum}_j) \quad (s = \text{sum}) \quad (2.4a)$$

Para el segundo método de fusión adoptaron una estrategia de toma de decisión basada en reglas de producción que son normalmente usados por sistemas expertos en modelado cognitivo e inteligencia artificial. Para esto combinan el set de datos obtenidos en un modelo condición del tipo.

$$R_i = SiP \Rightarrow Q \quad (2.5a)$$

Donde R_i representa la regla i, P es el antecedente de la regla i, y Q es el resultado de la regla i, representado por la siguiente tabla.

R_i	P	Q
R_1	(felicidad/fuerte)	Felicidad
R_2	(felicidad/moderado)	Felicidad
R_3	(felicidad/débil)	Neutral
R_4	(neutral/fuerte)	Felicidad
R_5	(neutral/moderado)	Neutral
R_6	(neutral/débil)	Neutral
R_7	(tristeza/fuerte)	Miedo
R_8	(tristeza/moderado)	Tristeza
R_9	(tristeza/débil)	Tristeza
R_{10}	(miedo/fuerte)	Miedo
R_{11}	(miedo/moderado)	Miedo
R_{12}	(miedo/débil)	Tristeza

Cuadro 2.1: Matriz de Reglas (R_i), Antecedentes (P) y Resultados (Q) para toma de decisiones

2.3.2. Conclusión trabajo de Luis Vergara, Yongrui Huang, Jianhao Yang, Pengkai Liao, and Jiahui Pan

El experimento realizado con 20 voluntarios arrojó que la precisión de la primera detección de fusión utilizando una regla de suma es del 81,25 % y la precisión de la segunda detección de fusión utilizando una regla de producción es del 82,75 %, ambas más altas que la de la expresión facial (74,38 %) o la detección de EEG (66,88 %).

2.4. Resumen

En este capítulo se ha informado respecto al estado del arte de la tecnología EEG + BCI. Se han analizado proyectos relevantes, que mantienen un grado de relación con la presente tesina, en donde se ha demostrado el alcance, impacto y falencias de la tecnología EEG frente a aspectos claves como tiempo de reacción y performance, nivel de precisión en las respuestas, capacidad de adaptación y versatilidad de la tecnología, a la vez que nos orienta en la realización de la presente tesina.

Posteriormente se detalla porque se orienta al uso y realización de tecnología open source para llevar adelante el proyecto.

Capítulo 3

Tecnologías Open Source

3.1. Introducción

Este capítulo pretende brindar un acercamiento breve y conciso a las tecnologías Open Source, necesario para comprender posteriormente en aquellas herramientas que se utilizarán específicamente en este proyecto.

3.2. ¿Qué son las tecnologías Open Source?

La expresión Open Source¹ hizo referencia en sus orígenes a un tipo de software cuya característica destacable es que sus creadores dejan a disponibilidad pública su código fuente bajo un tipo especial de licencia pensada para ese fin. Esto quiere decir que cualquier persona puede ver, modificar y distribuir ese código en la forma que crea conveniente.

Si bien esta fue la concepción original, la filosofía Open Source se convirtió en una forma de trabajo aplicable a otros ámbitos mas allá del software, pudiendo utilizarse también en la producción y utilización de hardware (Descripción elaborada a partir del sitio [5] de Red Hat Inc., líder mundial en Open Source).

3.3. Beneficios del software Open Source

Al día de hoy, el software open source ha penetrado en todo tipo de ámbitos de trabajo, desde estudiantes dando sus primeros pasos en el desarrollo de software, hasta corporaciones multinacionales que, al verse beneficiadas por este tipo de herramientas, han tomado una posición activa y han comenzado a colaborar en su mejora y también en la producción de herramientas propias, generando una retroalimentación que beneficia a todos los involucrados.

¹Open Source: Una traducción literal al español puede ser “fuente abierto” o “Código fuente abierto”, aunque la traducción utilizada es “Código Abierto”

En [6] se han detallado una serie de características que hacen del software open source, la herramienta ideal para el tipo de proyecto a implementar:

- **Costo nulo o más bajo que productos propietarios**

El espíritu del software libre se sustenta en el esfuerzo colaborativo y en muchos casos altruista, o con fines puramente intelectuales, por lo cual suele ser gratuito, o de muy bajo costo.

- **Problemas de seguridad esporádicos y de baja incidencia**

Al ser productos mantenidos por la comunidad, el software está sometido a un escrutinio constante, lo cual facilita la detección temprana de errores y la robustez de las herramientas.

- **Conocimiento compartido**

El software privativo se produce en espacios cerrados de trabajo, al que solo acceden los desarrolladores involucrados en el proyecto, lo cual limita las posibilidades de compartir el conocimiento, generando una visión sesgada del producto y sus posibilidades. El software open source abre las puertas a gran diversidad de puntos de vista para un mismo producto, producto de la comunidad que lo mantiene y su interacción con actores ajenos al proyecto para el cual fue ideado.

- **Adaptabilidad**

Todo producto de software libre posibilita la modificación, adaptación y expansión de su idea original. Excelentes productos de software de uso masivo en la actualidad se constituyen o se basan en productos de código abierto. Algunos de los casos más conocidos son los navegadores Chrome de Google o Firefox de Mozilla.

- **Visión de usuario** En general, los desarrolladores crean herramientas de código abierto con el fin de suplir las funcionalidades que ellos mismos requieren, lo cual les da una visión de usuario única y no replicable en otro tipo de proyectos.

- **Flexibilidad de licencias** El software open source se rige por una enorme cantidad de tipos de licencia, que posibilitan distintos niveles de restricción sobre el producto original, haciéndolo muy adaptable.

- **Disponibilidad**

Debido al mantenimiento comunitario de los productos de software open source, siempre que exista un equipo interesado en su mantenimiento, su disponibilidad es prácticamente perpetua. Incluso en caso de que el producto no reciba mantenimiento, el mismo puede seguir disponible para su utilización.

3.4. ¿Por qué utilizar tecnologías Open Source?

Una de las metas generales de este proyecto es generar herramientas de fácil accesibilidad y disponibilidad para el público en general, debido al enorme potencial que pueden tener los resultados obtenidos para la salud pública.

Con esto en mente se opta por la utilización de tecnologías de desarrollo Open Source. Este tipo de tecnologías incluye software y hardware de todo tipo, que se encuentra a disposición del público en general a bajo o nulo costo, con grandes comunidades que trabajan día a día para su mantenimiento y mejora.

3.5. Resumen

En este capítulo se ha hecho una muy breve descripción de las tecnologías Open Source, un repaso por los principales beneficios del software producido bajo esta filosofía y finalmente la motivación que nos ha llevado a optar por este tipo de herramientas. Todo esto como preámbulo para incursionar en la descripción de las herramientas específicas a utilizar para implementar la solución objetivo.

Capítulo 4

Software utilizado

4.1. Introducción

Este capítulo detalla las características generales de las herramientas y lenguajes de programación utilizadas para la implementación de la solución propuesta.

4.2. JavaScript

El paper [7] nos da el contexto suficiente para entender que en la última década el lenguaje de programación JavaScript ha cobrado una relevancia sin precedentes que crece continuamente, involucrándose con todo tipo de entornos de ejecución, permitiendo desarrollar aplicaciones para navegadores web, servidores web basados completamente en este lenguaje, e incluso aplicaciones de escritorio.

Gracias a esto ha surgido una gran variedad de herramientas de desarrollo para todo tipo de fines. Estas herramientas son respaldadas tanto por comunidades de desarrolladores independientes, como por corporaciones multinacionales, mejorando y potenciando su mantenimiento y desarrollo.

Este auge nos permite contar con grandes cantidades de información, alternativas de herramientas plenamente documentadas y ejemplos desarrollados por una comunidad muy activa, que genera contenido mayormente de acceso público, en la forma de software Open Source. En este contexto JavaScript resalta por sobre otros lenguajes por su gran versatilidad.

4.2.1. Impacto

JavaScript es el lenguaje de programación más utilizado para aplicaciones del lado del cliente, con un 97% de cuota de mercado, frente a otros lenguajes. En aplicaciones del lado del servidor JavaScript cuenta con una modesta cuota de mercado del 1.4%, quedando en octavo lugar, pero teniendo en consideración que en el corto

tiempo que ha pasado desde que comenzó a utilizarse como lenguaje de servidor de forma masiva, ha tenido un crecimiento muy destacable.¹

4.2.2. Entorno

Uno de los grandes beneficios de JavaScript, y un desafío de igual manera, es la gran versatilidad que ha obtenido en años recientes, pudiendo emplearse prácticamente en todo tipo de entorno. Las aplicaciones desarrolladas en JavaScript pueden ejecutarse en arquitecturas Cliente-Servidor, tanto del lado del cliente (En navegadores web), como del lado del servidor (Principalmente mediante Node.js, el cual será tratado a continuación), pero también pueden ejecutarse en entornos de escritorio mediante herramientas como Electrón.

4.2.3. Rendimiento

Aunque los límites entre lenguaje interpretado y compilado, en la actualidad ya no están tan definidos como en sus orígenes, JavaScript es un lenguaje interpretado, por lo que no puede igualar la velocidad de ejecución de lenguajes compilados tales como C++ o Java. Por cuestiones de seguridad, JavaScript tampoco tiene la capacidad de acceder de forma directa a unidades de GPU. Estas son limitaciones de peso al momento de pensar en JavaScript como alternativa de desarrollo.

Para contrarrestar las complicaciones de base del lenguaje, han emergido ciertos estándares. Uno de ellos es WebAssembly (Wasm), un formato de instrucciones binarias para máquinas virtuales, diseñado como destino de compilación portable para lenguajes de programación, lo que permite el despliegue aplicaciones para cliente y servidor en la web. De esta manera, un navegador web puede interpretar y ejecutar de forma directa programas escritos en C++, lo cual permite en ciertas situaciones superar el rendimiento de JS. Estos detalles y más pueden investigarse en mayor detalle en el sitio web [8].

Otra alternativa es WebGL, que según el sitio oficial [9], es una especificación estándar que define una API implementada en JavaScript para la renderización de gráficos en 3D dentro de cualquier navegador web. Esta API expone OpenGL a JavaScript. Esta API es Cross-Language y Cross-Platform y tiene como finalidad la renderización de vectores gráficos en 2D y 3D, y permite tareas de renderización de videojuegos y otro tipo de aplicaciones de alto rendimiento dentro de páginas web.

WebAssembly y WebGL abrieron posibilidades para el desarrollo de múltiples herramientas, algunas de las cuales se describirán más adelante en este documento.

Del lado del servidor, las librerías implementadas en JavaScript pueden conectarse con módulos nativos escritos en C y C++ mediante la N-API de Node.js.

¹Estadísticas tomadas del sitio w3techs.com, dedicado a encuestas de uso de tecnologías web

4.2.4. Compatibilidad Cross-Browser

JavaScript está diseñado para ser un lenguaje Cross-Browser, y está soportado por los navegadores web más reconocidos, con APIs estandarizadas que facilitan el desarrollo de aplicaciones compatibles con todas las plataformas.

4.2.5. Ejecución mono hilada

JavaScript es un lenguaje de naturaleza mono hilada, es decir que cuenta con un hilo de ejecución principal. Esto simplifica el desarrollo de aplicaciones, pero implica la responsabilidad de no bloquear el hilo de ejecución, para no impactar en el rendimiento de la aplicación en su conjunto, manteniendo un balance en el uso de APIs sincrónicas y asíncronas.

4.3. Node.js

“Node.js” es un entorno de ejecución basado en el lenguaje de programación JavaScript, construido sobre el motor “V8”, conocido también por ser el motor base para el navegador web Chrome desarrollado por Google, inc.

4.3.1. Características

El sitio oficial de Node.js, [10], nos indica sus características principales. Node.js conforma un entorno de ejecución de JavaScript asíncrono guiado por eventos, diseñado para construir aplicaciones de red escalables, sin depender de la utilización de hilos de ejecución del sistema operativo en el que se ejecuta.

El modelo de eventos de Node.js presenta un ciclo de eventos como una construcción de tiempo de ejecución y no como una librería invocable. En otros sistemas se requiere una llamada de bloqueo para iniciar el ciclo de eventos, y generalmente el comportamiento se define mediante callbacks al comienzo de un script de ejecución. Luego, al finalizar, se inicia un servidor mediante otra llamada de bloqueo.

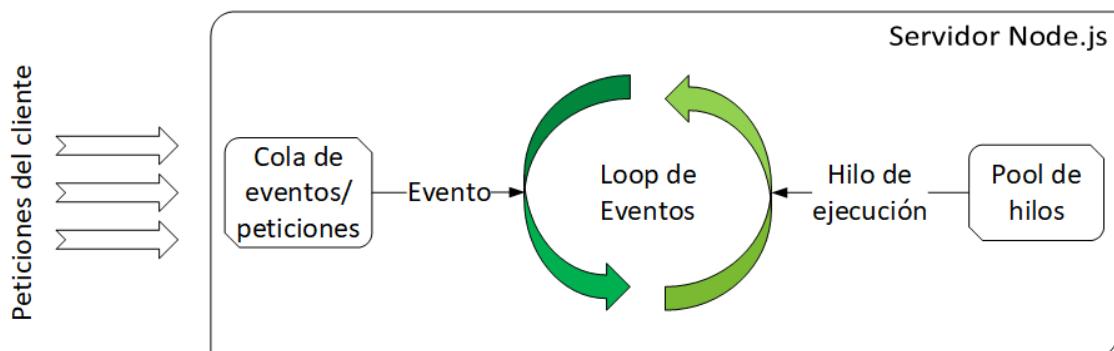


Figura 4.1: Arquitectura de Node.js

En Node.js el comportamiento es distinto. En primer lugar se ejecuta un script de entrada, que inicia un ciclo de eventos de finalización indefinida. El funcionamiento de Node.js se puede ver reflejado en la figura 4.1. El loop de eventos estará encargado de ejecutar todos los trabajos y callbacks pendientes originados a partir de peticiones de entrada. Estas peticiones se acumulan en una cola de espera. Por otro lado, se cuenta con un pool de hilos de ejecución disponibles para la ejecución de trabajos. Siempre que se cuente con un hilo de ejecución libre, se podrá tomar una petición de la cola de espera, para que esta ingrese en el loop de eventos. Todo esto se ejecuta de forma asíncrona y así el loop de eventos logra quedar oculto al usuario.

El entorno de ejecución también es libre de bloqueos, debido a que la gran mayoría de las funciones disponibles no ejecutan operaciones de entrada/salida de forma directa. Esta característica hace de Node.js un entorno muy favorable para el desarrollo de aplicaciones escalables.

Node.js permite aprovechar procesadores de múltiples núcleos, a pesar de estar diseñado para no usar hilos de ejecución. Para ello cuenta con una API de ejecución de procesos hijos. Estos procesos también pueden comunicarse entre sí, gracias al módulo de cluster, que permite compartir sockets entre procesos para habilitar el balance de núcleos de procesamiento.

4.4. Firestore

Firestore es una base de datos NoSQL flexible, escalable y en la nube. Tiene el fin de almacenar y sincronizar datos para el desarrollo tanto del lado del cliente como del servidor.

Se centra en la creación de documentos que contienen campos a los cuales se le asignan valores. Estos documentos se almacenan en colecciones, que son contenedores para los documentos y que se utilizan para organizar los datos y compilar consultas.

Los documentos admiten varios tipos de datos diferentes, desde strings y números simples, hasta objetos anidados complejos. También permite crear subcolecciones dentro de documentos y crear estructuras de datos jerárquicas que se ajustan a escala a medida que la base de datos crece. Aparte de la flexibilidad y escalabilidad que se puede lograr con este tipo de base de datos, el modelo de datos de Cloud Firestore admite cualquier estructura de datos.

4.5. TensorFlow.js

TensorFlow es una plataforma de código abierto flexible que permite crear algoritmos de machine learning. Sus implementaciones pueden ejecutarse sin cambios en gran variedad de entornos y dispositivos tales como teléfonos, tablets, e incluso sistemas distribuidos a gran escala, mediante lenguajes de programación tales como Python, C++, Java y más recientemente JavaScript.

Particularmente TensorFlow.js (TFJS) es una librería especial que permite ejecutar modelos de TensorFlow en navegadores web y en el entorno de servidor provisto por Node.js. Esta librería provee un conjunto de APIs compatibles con las APIs de Python, por lo cual los modelos son portables entre ambos lenguajes. Así se pueden generar muchas oportunidades en entornos browser-based:

- **Universalidad:** TFJS se diseñó con el objetivo de transportar la versatilidad de TensorFlow a navegadores web estándar, y así acceder a un público mucho más amplio disminuyendo la barrera de ingreso a las técnicas de Machine Learning.
- **Interactividad:** La naturaleza interactiva de los navegadores web y la versatilidad de las APIs Web facilita en gran medida el desarrollo de aplicaciones de Machine Learning centradas en el usuario.
- **Acceso a periféricos:** Usando el navegador como intermediario, es posible acceder a mediciones de sensores mediante APIs estandarizadas. Así, al desarrollar se pueden utilizar periféricos como cámaras web, micrófonos e incluso acelerómetros, de manera relativamente sencilla.

Estas características convierten a TFJS en una excelente herramienta de divulgación de conocimiento, que puede usarse tanto para propósitos educativos como para desarrollo de aplicaciones para entornos de producción. La información referida al funcionamiento de TFJS puede explorarse con mayor detalle en el paper [7].

4.5.1. Arquitectura de TensorFlow.js

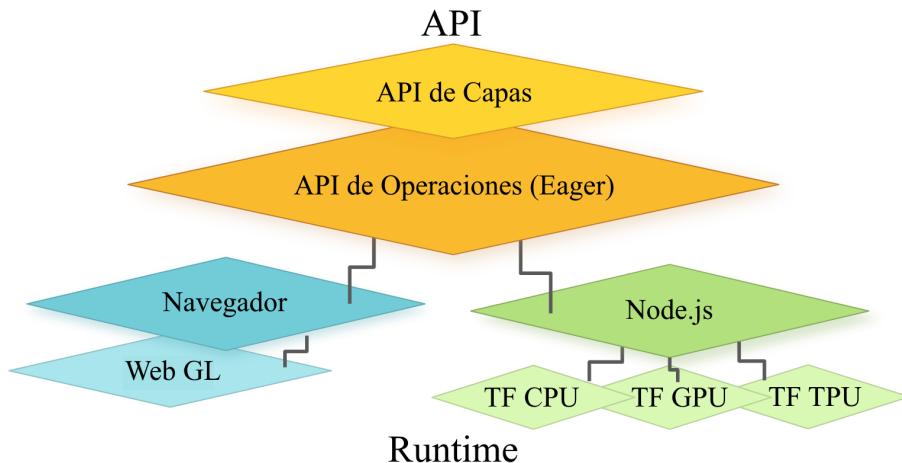


Figura 4.2: Arquitectura de TensorFlow.js, extraído y traducido a partir de [7].

La figura 4.2 nos da un vistazo de la arquitectura a alto nivel de TFJS. En principio se cuenta con dos APIs generales: La API de capas (Provee bloques de construcción de modelos a mas alto nivel además de buenas prácticas con énfasis en redes neuronales) y la API de operaciones (Provee operaciones de álgebra lineal de bajo nivel). La API de capas esta modelada a partir del namespace `tf.keras` de

la versión original de TensorFlow escrita en Python, que a su vez esta basada en la API Keras ².

Como se mencionó anteriormente, TFJS esta diseñado para ejecutarse en navegadores y también del lado del servidor. En el navegador, utiliza la GPU del dispositivo mediante WebGL, para habilitar cálculos de punto flotante paralelizado. En el lado del servidor, mediante Node.js, la versión JavaScript tiene acceso a una librería en C que habilita acceso completo a la versión de TensorFlow original, escrita en lenguaje Python. Estas alternativas pretenden sacar el máximo provecho del dispositivo utilizado, pero si TFJS no llegara a tener acceso a WebGL o la librería C, puede ejecutarse en cualquier ambiente mediante JavaScript puro, por supuesto que a una velocidad menor por las limitaciones del lenguaje.

4.5.2. Integración en el ecosistema de TensorFlow

La arquitectura de TensorFlow.js permite importar y convertir modelos de TensorFlow de distintas plataformas, en modelos para TFJS. Existe un enorme catálogo oficial de ejemplos y modelos pre-entrenados de los cuales nutrirse.

Con el fin de allanar el camino para desarrolladores e investigadores principiantes, TFJS cuenta con APIs especiales de alto nivel para ocultar los detalles más complejos y técnicos, permitiendo trabajar con elementos más conocidos en el mundo del desarrollo web, como elementos del DOM, arrays primitivos u objetos de JavaScript.

4.6. Vue.js

Vue.js es un framework de desarrollo web front-end, que permite construir aplicaciones, utilizando una estructura de componentes y el paradigma de programación reactiva.

4.6.1. Estructura de componentes

Un componente es una estructura instanciable, compuesta por código JavaScript, CSS, y HTML. Implementando este concepto, un sitio web puede construirse como una estructura de componentes dentro de componentes, donde el sitio o aplicación web es, en sí mismo, un componente más y también el de más alto nivel como se muestra en la figura 4.3

²La API Keras es una API escrita en lenguaje Python, utilizada para construir redes neuronales.

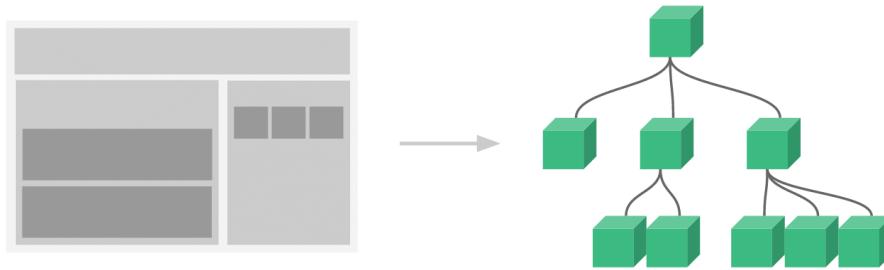


Figura 4.3: Estructura de árbol de componentes de Vue.js, extraído de [11].

4.6.2. Programación reactiva

Este paradigma permite desarrollar aplicaciones basadas en flujos de datos y respuestas a eventos, y de esa manera construir una interfaz de usuario que reacciona automáticamente a las modificaciones de la información de cada componentes, así el desarrollador no necesita preocuparse por mantener la interfaz actualizada. Los tiempos de desarrollo se reducen reutilizando estructuras comunes, y la construcción de aplicaciones dinámicas se facilita enormemente.³

En la figura 4.4 se explica de forma básica el comportamiento de los componentes en Vue.js. El patrón de diseño Observador permite a cada componente ser notificado cuando ocurren cambios de datos, lo que provoca una reacción en cadena que provoca el re-renderizado de todos los componentes del árbol, manteniendo la aplicación permanentemente actualizada.

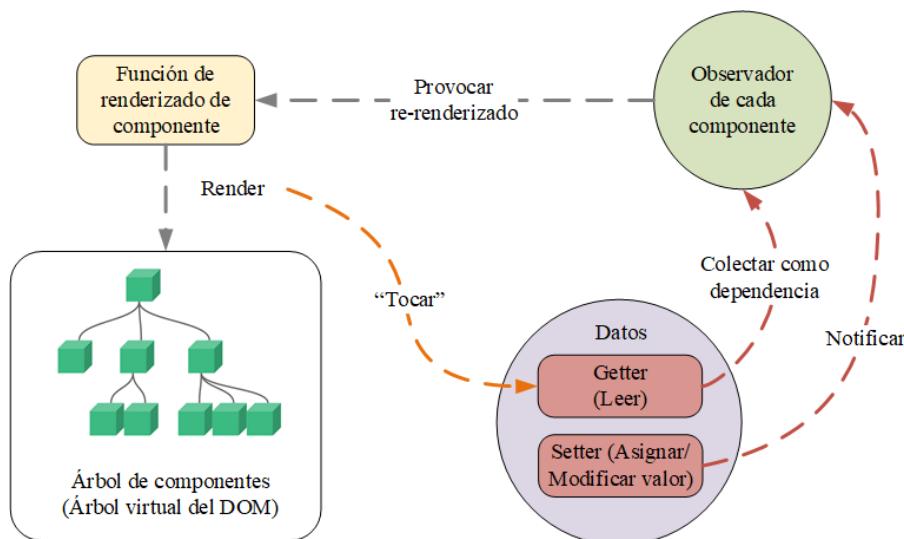


Figura 4.4: Reactividad y ciclo de actualización de componentes, extraído de [11].

4.6.3. Accesibilidad

Por lo descrito por su propio creador en [12], este framework fue concebido con la idea de ser accesible, con una curva de aprendizaje empinada, que permita lograr en poco tiempo grandes resultados. Siguiendo esta filosofía, el framework evolucionó

³Descripción de autoría propia basada en experiencias personales con la herramienta

para convertirse en una herramienta de fácil aprendizaje y que puede adoptarse de manera incremental. Estas características provocan que sus creadores lo definan como un “Framework progresivo”.

4.7. Resumen

En este capítulo se ha hecho un repaso y descripción de las herramientas de software ha utilizar en este proyecto, indagando en sus características mas importantes con el detalle suficiente como para comprender la arquitectura sobre la que se construyen y su utilidad en múltiples entornos de aplicación.

A continuación se analizan las características del hardware utilizado para llegar a la solución propuesta.

Capítulo 5

Hardware utilizado

5.1. Introducción

La medición electroencefalográfica (EEG) es la medida de la actividad cerebral. El primer acercamiento a dicha medida fue registrado por Hans Berger(1924)¹ usando un simple galvanómetro, que es básicamente un instrumento que se usa para detectar y medir la corriente eléctrica. En el cuero cabelludo humano se colocó solo un electrodo y se identificó una onda. Era una onda α (alfa, también llamada onda de Berger). Recientemente se han desarrollado dispositivos EEG económicos gracias a la inversión de distintas marcas como Avatar EEG solutions, Neurosky inc., OCZ Technology, InteraXon, PLX Devices y Emotiv Systems entre otros. Estas soluciones desarrolladas de bajo costo actualmente no son implementadas por las ciencias médicas, pero si muy adoptadas para interfaces BCI de uso convencional.

¹Hans Berger fue un neurólogo y psiquiatra alemán, considerado en la actualidad el padre de la electroencefalografía por sus estudios pioneros sobre la actividad eléctrica en el cerebro humano

5.2. Mindwave MW001

Uno de los dispositivos EEG más baratos del mercado es el modelo “MindWave MW001” de un canal producido por Neurosky Inc.

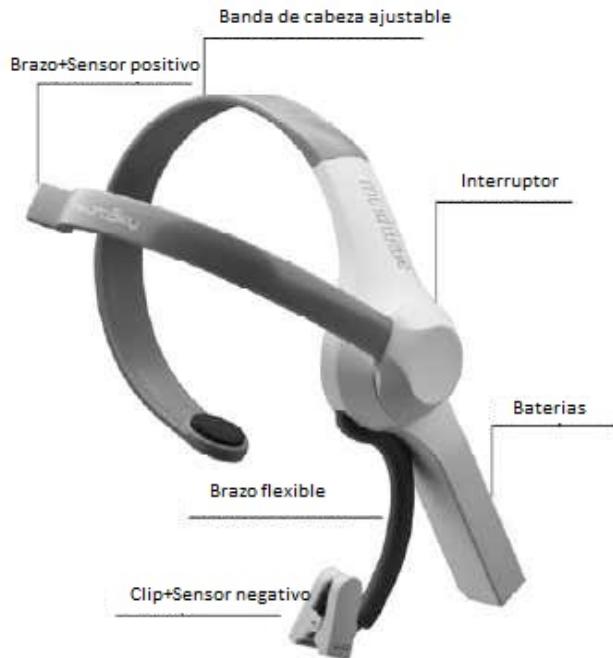


Figura 5.1: MW001

5.2.1. Composición

El dispositivo 5.1 consta de ocho partes principales, clip para la oreja, brazo flexible, área de pila 1.5v AAA, interruptor de encendido, banda para la cabeza ajustable, punta del sensor, brazo del sensor y el chipset ThinkGear interno.

5.2.2. Funcionamiento

El principio de funcionamiento es bastante sencillo. Se utilizan dos sensores secos para detectar y filtrar las señales de EEG. La punta del sensor detecta señales eléctricas de la frente del cerebro. Al mismo tiempo, el sensor capta el ruido ambiental generado por músculos humanos, computadoras, bombillas, enchufes eléctricos y otros dispositivos eléctricos. El segundo sensor, clip para la oreja, es una conexión a tierra y una referencia, que permite que el chip ThinkGear filtre el ruido eléctrico.

El dispositivo mide:

- La señal sin procesar
- El espectro de potencia (alfa, beta, delta, gamma, theta)

- El nivel de atención
- El nivel de meditación
- La detección de parpadeo

Los datos brutos del EEG se reciben a una frecuencia de 512 Hz, y los datos emitidos están dados en:

$$Resultado = \frac{[alfa, beta, delta, gamma, theta]}{Tiempo} \quad (5.1)$$

Para su funcionamiento se debe tener instalado la dll provista por el fabricante para vinculación dinámica con el dispositivo (thinkgear.dll).

5.2.3. Comunicación

Los datos del dispositivo son recibidos mediante el protocolo Bluetooth RF-COMM que es conjunto simple de protocolos de transporte, construido sobre el protocolo L2CAP; y que proporciona sesenta conexiones simultáneas para dispositivos Bluetooth emulando puertos serie RS-232. El protocolo está basado en el estándar ETSI TS 07.10.

5.2.4. Protocolo

ThinkGear Socket Protocol (TGSP) es un protocolo basado en JSON para la transmisión y recepción de datos de ondas cerebrales entre cliente y servidor.

Está diseñado para permitir que los lenguajes del lado del servidor puedan leer de manera correcta datos del puerto serie estándar y así logren integrar fácilmente la funcionalidad de detección de ondas cerebrales a través de las API de socket.

Para lograr conectar el dispositivo bajo este protocolo se debe.

- Crear un socket de conexión
- Autorizar el dispositivo cliente en el servidor (única vez)
- Configurar parámetros del servidor
- Recibir y procesar los datos
- Terminar la conexión del socket

En la imagen 5.2 se puede ver representado el protocolo implementado.

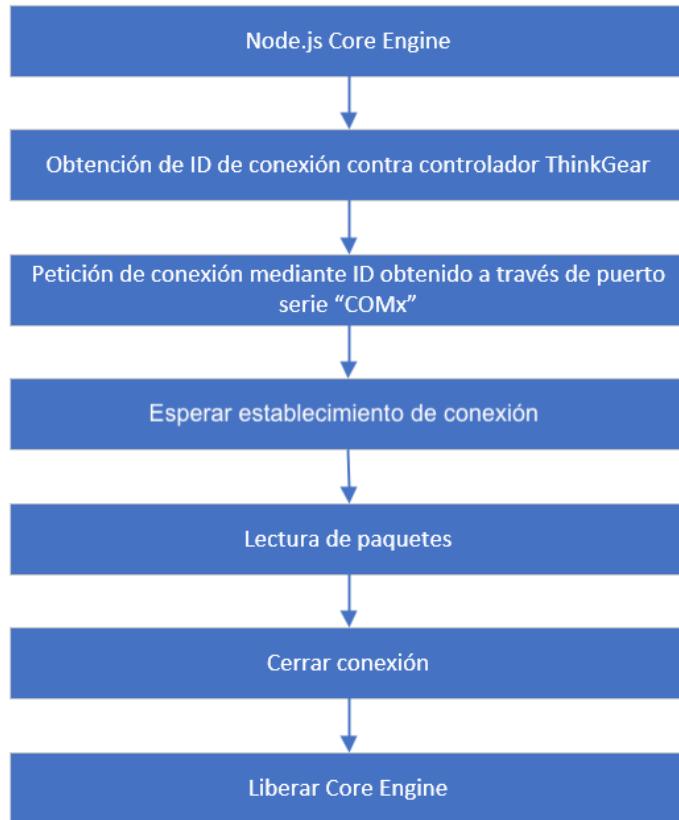


Figura 5.2: Adaptación de protocolo a solución propuesta

5.3. Hardware para pruebas

Para desarrollo y pruebas se utilizó principalmente una computadora portátil con 12GB de memoria RAM y un procesador AMD Ryzen 7 3700U de 4 núcleos y 2.30 GHz. Esta computadora cuenta con un sistema operativo Windows 10 de 64 bits. Adicionalmente para desarrollos y algunas pruebas de conexión con el casco EEG se utilizó una segunda computadora portátil con 12GB de memoria RAM y un procesador AMD HexaCore de 4 núcleos de 2.0 GHz (La denominación HexaCore contempla 2 núcleos de GPU). Estos equipos se utilizaron por disponibilidad, más que por una demanda específica de la solución propuesta.

5.4. Resumen

En este capítulo se ha revisado el hardware utilizado para llevar a cabo el desarrollo. Detallando en base a sus especificaciones sus cualidades, funcionamiento, forma de comunicación y protocolos vigentes, mostrando el flujo de trabajo de la adaptación propuesta del protocolo del fabricante al desarrollo. También se menciona muy brevemente el hardware utilizado para desarrollo de software y prueba de la solución.

A continuación se hará una breve introducción a los temas de inteligencia artificial y machine learning.

Capítulo 6

Inteligencia Artificial y Machine Learning

6.1. Introducción

El uso de la Inteligencia Artificial (IA) está cada vez más presente en nuestro día a día. En [13], investigadores de la Universidad de Stanford concluyen que pueden esperarse incrementos sustanciales en los usos futuros de aplicaciones de IA, entre las que pueden incluirse el aumento de los vehículos autónomos, diagnósticos y tratamientos médicos especializados, y asistencia para ciudadanos de la tercera edad. Este reporte también considera que la sociedad se encuentra en una coyuntura crucial para determinar cómo desplegar tecnologías basadas en IA de manera de que promuevan valores tales como la libertad, la igualdad y transparencia.

En [13] se destacan principalmente los siguientes datos importantes en relación a la IA:

- **Publicaciones:** La cantidad de papers publicados que están enfocados en IA muestran un aumento sostenido entre los años 2010 y 2019, llegando a contabilizar en torno a las 20000 por año. Destaca la popularidad de las investigaciones sobre Machine Learning (En el sitio arXiv.org¹, la cantidad de este tipo de publicaciones se ha duplicado cada año desde 2009 a 2017).
- **Estudiantes:** La inscripción en cursos relacionados a IA se ha visto incrementada 16 veces a nivel internacional tomando como punto de comparación el año 2010. Esto convierte a la especialización en IA, en la más popular dentro de las ramas de la informática.
- **Velocidad:** El volumen de poder de cómputo utilizado en aplicaciones de primer nivel se duplica cada 3.4 meses. Lo cual implica, entre otras cosas, que el entrenamiento de modelos de Machine Learning requiere cada vez menos tiempo.

¹arXiv: Servicio de distribución y archivo de acceso público gratuito que alberga casi dos millones de artículos académicos, propiedad de Cornell University.

- **Comparación con capacidades humanas:** Para 2019, los sistemas de inteligencia artificial han superado el rendimientos de nivel humano en actividades tales como reconocimiento de objetos mediante ImageNet², reconocimiento de lenguaje en dominios limitados, traducción de Chino a Inglés en dominios restringidos, detección de cánceres de próstata y piel, diagnóstico de retinopatía diabética, e incluso juegos como el ajedrez, el juego de estrategia Go, poker, el juego de preguntas y respuestas Jeopardy! y videojuegos como el clásico Pac-Man, Quake III, Dota 2, StarCraft II, y múltiples videojuegos de la clásica consola Atari.

6.1.1. Aplicaciones actuales

Los siguientes son algunos de los ejemplos de aplicaciones de inteligencia artificial en la actualidad:

- **Vehículos robóticos:** Los primeros eran controlados mediante radio control durante los años 20 del siglo XX. A partir de los años 80 comenzaron a surgir los primeros indicios de autonomía. En el año 2018 los vehículos de prueba de la compañía Waymo superaron la barrera de las 10 millones de millas conducidas en caminos públicos sin accidentes serios, solo requirieron la intervención de un humano una vez cada 6000 millas. Poco tiempo después, la compañía comenzó a ofrecer un servicio comercial de taxi robótico. En cuanto a vehículos aéreos, Rwanda ha estado utilizando drones autónomos para hacer entregas de sangre desde 2016. Los cuadrocópteros son capaces de efectuar elaboradas maniobras acrobáticas, explorar edificios al mismo tiempo que construyen mapas 3D, e incluso construir formaciones autónomas
- **Medicina:** Un énfasis recurrente en la inteligencia artificial médica es la colaboración humano-máquina. Actualmente los algoritmos logran igualar o exceder a doctores expertos en el diagnóstico de condiciones, especialmente en diagnósticos basados en imágenes. La adopción generalizada de estas técnicas se encuentra limitada no por la precisión de diagnóstico sino por la necesidad de demostrar mejoras en los resultados y asegurar transparencia, mínimo sesgo y privacidad de datos. En 2017, solo dos aplicaciones de inteligencia artificial fueron aprobadas por la Food and Drugs Administration de Estados Unidos. Se incrementó a 12 en 2018 y más en años siguientes.

En años recientes la popularidad del uso de la Inteligencia Artificial, y del Machine Learning como campo de estudio y trabajo ha crecido exponencialmente, debido principalmente, al abaratamiento de costos de hardware de alto poder de cálculo y al surgimiento, en consecuencia, de herramientas de características open source de fácil uso y al alcance de la mano de investigadores de todo tipo de nivel académico y profesional.

²El proyecto ImageNet es una gran base de datos visual diseñada para utilizar en investigaciones de software de reconocimiento de objetos.

6.2. Machine Learning

El Machine Learning (Aprendizaje de máquina por su traducción al español) constituye una disciplina que se ha vuelto fundamental dentro del campo la inteligencia artificial, y cómo tal, puede estudiarse de forma específica y determinar qué actividades engloba.

Es una disciplina por la cual se logra proveer a una máquina de las herramientas necesarias para que desarrolle la capacidad de responder de forma lógica y esperable a un impulso específico. La maquina logra desarrollar esta capacidad mediante la observación de ejemplos concretos del problema que se pretende resolver. Esta capacidad implica que la maquina podrá de obtener información, procesarla, analizarla y en base a los resultados obtenidos, mejorar su proceso de toma de decisiones de forma progresiva, aprendiendo de su propio comportamiento y decisiones pasadas.

En [14] se cuestiona la necesidad de proveer a una máquina de tal capacidad, pero inmediatamente se plantean dos grandes motivos por los cuales es una característica deseable. En primer lugar, uno podría pensar que sería más sencillo diseñar un sistema orientado específicamente a resolver el problema que se nos presenta, sin embargo, la realidad nos demuestra continuamente que es prácticamente imposible para un diseñador anticipar todos los escenarios que podrían presentarse, por lo que el sistema necesitará ser capaz de reaccionar a situaciones imprevistas. En segundo lugar, muchas veces ocurre que los diseñadores no pueden llegar a una solución adecuada por sus propias limitaciones. Cuando esto ocurre es mucho más conveniente utilizar algoritmos de Machine Learning, y que sea la maquina la que encuentre una solución. Por ejemplo, una persona sabe reconocer ciertos patrones de imágenes, pero ese es un comportamiento difícil de replicar.

6.2.1. Aprendizaje

Podemos decir en palabras muy simples que aplicando Machine Learning, una maquina puede partir desde un conjunto de observaciones y llegar a una regla general que puede aplicarse reiteradamente para obtener resultados. Este proceso se conoce como inducción y es propenso a errores, por lo que difiere del proceso conocido como deducción, en el que se logra llegar a resultados correctos siempre, a menos que la premisa sea incorrecta.

Los problemas a resolver mediante Machine Learning reciben dos denominaciones posibles. Cuando el resultado a obtener es uno entre un conjunto finito de posibles valores, el problema es conocido como problema de “Clasificación”. Cuando el resultado a obtener es un número, entero o real, la denominación es problema de Regresión.

El tipo de aprendizaje a emplear estará determinado por el tipo de entrada o fuente de datos a utilizar. La teoría del Machine Learning nos presenta tres tipos principales de aprendizaje:

6.2.1.1. Aprendizaje no supervisado

En este caso los algoritmos descubren patrones ocultos en datasets no etiquetados sin la necesidad de intervención humana. Esta capacidad de descubrir similitudes y diferencias en los datos, hacen que sea el tipo de aprendizaje ideal para análisis de datos exploratorio, segmentación de datos o reconocimiento de imágenes y patrones. También se utiliza para reducir el número de características en un modelo mediante el proceso de reducción dimensional; el análisis de componentes principales (PCA: Principal Component Analysis) y descomposición de valores singulares (SVD: Singular Value Decomposition) son dos aproximaciones comunes a esto. Otros algoritmos usados en el aprendizaje no supervisado incluyen redes neuronales, agrupamiento K-means, métodos de agrupamiento probabilístico, y más.

6.2.1.2. Aprendizaje por refuerzo

Es un tipo de aprendizaje similar al supervisado, pero el algoritmo no es entrenado usando datos de muestra. Este tipo de modelos aprende siguiendo una serie de recompensas o castigos. Cada vez que se ejecuta una acción valorada como correcta, se recibe una recompensa. Luego el agente deberá determinar cuál de las acciones efectuadas fueron las que provocaron la recompensa o castigo.

Un buen ejemplo es el sistema Watson de IBM, que ganó el desafío del programa de concursos estadounidense “Jeopardy!” en 2011. El sistema utiliza este tipo de aprendizaje para decidir si debe intentar dar una respuesta (O pregunta), qué opción seleccionar en el tablero y cuánto apostar en el juego.

6.2.1.3. Aprendizaje supervisado

Depende de la utilización de datasets etiquetados para entrenar algoritmos y que de esta manera clasifiquen datos o predigan resultados de forma acertada. Se le asigna a los datos un peso o importancia. Conforme los datos de entrada son entregados al modelo, este corrige sus pesos hasta que el modelo se ajusta lo suficiente. Esto ocurre como parte del proceso de validación cruzada para asegurar que el modelo no compense demasiado o subestime sus resultados. De esta manera se logra crear una función que genera valores de salida apropiados de acuerdo a los valores de entrada.

Algunos métodos usados en aprendizaje supervisado incluyen redes neuronales, clasificadores bayesianos ingenuos, regresión lineal, regresión logística, árboles de decisión, bosques random, máquinas de vectores de soporte, entre otros.

Existe un punto intermedio entre aprendizaje no supervisado y aprendizaje supervisado. En el **aprendizaje semi-supervisado**, durante el entrenamiento, se utiliza un dataset etiquetado reducido, para luego guiar la clasificación y extracción de características a partir de dataset mayor no etiquetado. Este subtipo de aprendizaje puede resolver el problema de no tener suficientes datos etiquetados (o recursos suficientes para etiquetar datos) para entrenar un algoritmo de aprendizaje supervisado.

Si consultamos el libro [14], podremos ver que, de manera formal, la premisa del aprendizaje supervisado es la siguiente:

Dado un dataset de N elementos que tiene la forma

$$(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$$

dónde cada par es generado por una función desconocida f , descubrir una función h , llamada hipótesis, que aproxime la función real.

Esto nos indica que la función h representa una hipótesis que extraemos de un cierto espacio de hipótesis, el cuál contendrá una cantidad N de funciones alternativas. Informalmente, f representa nuestro modelo de aproximación al problema e y representa el valor real que pretendemos predecir con el, mediante los parámetros de entrada representados por x .

La elección del espacio de hipótesis dependerá de nuestro nivel de conocimiento respecto del dominio del problema planteado. En este sentido, se puede llegar a una hipótesis válida a través de 3 vías:

- Conocimiento previo del proceso que generó los datos.
- Análisis exploratorio de datos: Utilizar herramientas estadísticas a fin de encontrar una hipótesis que resulte suficientemente apropiada.
- Probar múltiples hipótesis y seleccionar aquella cuyos resultados tengan el mejor ajuste.

Al determinar cual será nuestra hipótesis es importante destacar que no se busca una función cuya precisión sea del 100 %, sino una que logre ajustarse lo más posible a los resultados deseados, considerando que encontrar una función de tal precisión no es una meta realista. Debido a esto, se debe entender que se tiene que observar cuantas predicciones son correctas a partir de un dataset de prueba, sino como se comporta la función frente a datos nuevos, nunca antes vistos.

Cuando se habla de precisiones y determinar la efectividad del modelo, se debe incorporar dos conceptos clave. Se sabe que el modelo no es 100 % preciso, por lo que se debe considerar la existencia de una medida de **Sesgo** o **Error** (“**Bias**” en inglés), la cual no es mas que la tendencia de nuestra hipótesis predictiva a alejarse de los resultados esperados, cuando se utiliza sobre diferentes datasets de entrenamiento, y que al mismo tiempo esta determinada por las propias restricciones del espacio de hipótesis. El segundo concepto a considerar es la **Variación** o **Varianza**, la cual representa el volumen de cambio presente en la hipótesis, provocada por la fluctuación en los datos de entrenamiento.

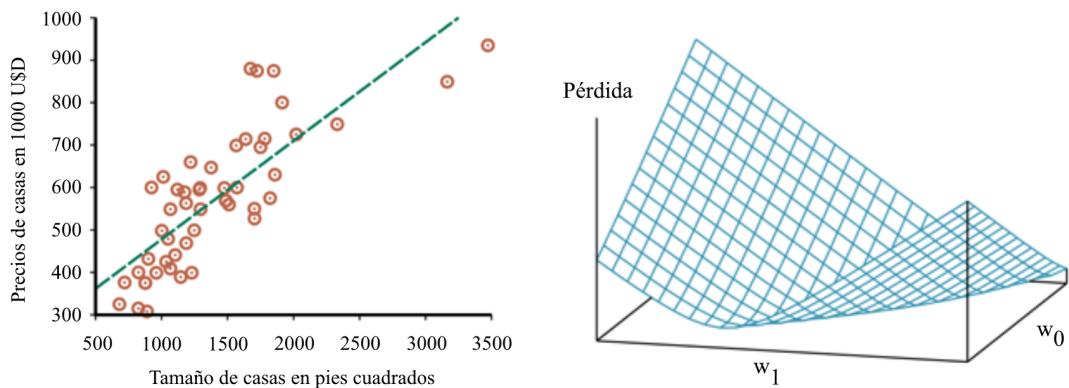
Considerando estos conceptos, podemos definir lo que se conoce como “**Ajuste**”, mencionado anteriormente (**Fitting** por su traducción al inglés). El ajuste tiene un umbral de utilidad. Un ajuste demasiado alto (Sobre ajuste u overfitting) provocará que la función hipótesis preste demasiada atención al dataset particular con que se está entrenando al modelo. Un ajuste demasiado bajo (Sub ajuste o Underfitting) provocará que la función hipótesis sea incapaz de encontrar patrones en los datos.

6.2.2. Regresión lineal

En base a lo expuesto en [15] en forma general, la regresión lineal es un modelo estadístico/matemático usado para la aproximación de la relación de dependencia entre una variable dependiente y , una cantidad m de variables independientes x , con $m \in \mathbb{Z}^+$ y un término aleatorio ϵ .

6.2.2.1. Regresión lineal simple

Una función lineal de una única variable (una línea recta) con entrada x y salida y tiene la forma $y = w_1x + w_0$ dónde w_0 y w_1 son coeficientes de tipo real que deben ser descubiertos. La letra w de los coeficientes hace referencia al peso; el valor de y es cambiado al cambiar el peso relativo de uno u otro de los términos. Definimos a w como el vector (w_0, w_1) y se define la función lineal con esos pesos como $h_w(x) = w_1x + w_0$. Para ajustar una linea a los datos, se requiere encontrar los valores de w_0, w_1 que minimicen las pérdidas, de acuerdo a la respectiva función de pérdida.



- (a) Gráfico de precios contra espacio habitable de una casa, responden a la función lineal de hipótesis que minimiza pérdidas por error al cuadrado: $y = 0,232x + 246$. Extraído de [14].
- (b) Función de perdida $\sum_j (y_i + w_1 x_j + w_0)^2$ para varios valores de w_0, w_1 . Notar que la función de perdida es convexa con un único mínimo global. Extraído de [14].

Figura 6.1: Ejemplo de regresión lineal, función lineal y función de pérdida, sobre un set de entrenamiento de n puntos en el plano x, y , cada punto representando relación entre tamaño en pies cuadrados y precio de un conjunto de casas en venta.

Generalmente se utiliza, como función de pérdida, la sumatoria de la función de error al cuadrado sobre todas las muestras de entrenamiento.

$$Perdida(h_w) = \sum_{j=1}^N L_2(y_i, h_w(x_j)) \quad (6.1)$$

$$= \sum_{j=1}^N (y_j - h_w(x_j))^2 \quad (6.2)$$

$$= \sum_{j=1}^N (y_i - (w_1 x_j + w_0))^2 \quad (6.3)$$

Pretendemos encontrar $w^* = argmin_w Perdida(h_w)$ ³.

La sumatoria vista en la formula 6.3 se minimiza cuando sus derivadas parciales respecto de w_0 y w_1 son cero:

$$\frac{\partial}{\partial w_0} \sum_{j=1}^N (y_i - (w_1 x_j + w_0))^2 = 0 \quad (6.4)$$

$$\frac{\partial}{\partial w_1} \sum_{j=1}^N (y_i - (w_1 x_j + w_0))^2 = 0 \quad (6.5)$$

Estas ecuaciones tienen una única solución:

$$w_1 = \frac{N(\sum x_j y_i) - (\sum x_j)(\sum x_i)}{N(\sum x_j^2) - (\sum x_j)^2} \quad (6.6)$$

$$w_0 = \frac{(\sum y_i - w_1(\sum x_i))}{N} \quad (6.7)$$

Muchas formas de aprendizaje involucran ajustar pesos para minimizar pérdidas, por lo que es útil tener una imagen de lo que sucede en el espacio de los pesos, el espacio definido por todas las posibles configuraciones de los pesos.

Para la regresión lineal de una variable, el espacio de pesos está definido por w_0 y w_1 es bidimensional, por lo que se puede graficar la pérdida como una función de w_0 y w_1 en un gráfico de 3 dimensiones. La consecuencia de esto es que la función de pérdida produce una representación convexa (Ver la figura 6.1(b)). Esto ocurre para cualquier regresión lineal con una función de pérdida, es decir que no existe mínimo local, sólo un mínimo general.

6.2.2.2. Regresión lineal multivariable

Es una extensión de la regresión lineal en la que x_j es un arreglo de n-elementos.

$$h_w(x_j) = w_0 + \sum_i w_i x_{j,i} \quad (6.8)$$

Podemos ver que en esta representación el término w_0 se presenta apartado del resto de los elementos de la sumatoria. Para solucionar esto puede utilizarse

³argmin: Punto del dominio de una función en la cual los valores de la función son minimizados

un término de la forma $x_{(j,0)} = 1$. De ésta manera la expresión puede reformularse como:

$$h_w(w_j) = \sum_i w_i x_{j,i} \quad (6.9)$$

El mejor vector de pesos, w^* , minimiza la pérdida de error al cuadrado:

$$w^* = \operatorname{argmin}_w \sum_j L_2(y_j, w \times x_j) \quad (6.10)$$

En cuanto al Gradiente Descendente, la función de actualización de cada peso w_i será

$$w_i = w_i + \alpha \sum_j (y_j - h_w(x_j)) \times x_{j,i} \quad (6.11)$$

6.2.3. Gradiente descendente

Este método permite minimizar las pérdidas y puede aplicarse a cualquier función de pérdida. Consiste en tomar como punto de comienzo cualquiera de los puntos del espacio de pesos, computar un estimado del gradiente y moverse de forma mínima en dirección decreciente. Este proceso deberá repetirse hasta converger en un punto en el espacio de pesos con un mínimo local de pérdidas. El tamaño del paso que se da en cada iteración es conocido como learning rate (Se puede traducir como “velocidad”, “ritmo” de aprendizaje). Puede ser una constante fija, o ir ajustándose conforme el proceso de aprendizaje avanza.

En el caso de la regresión lineal, la función de pérdida es cuadrática (Ver la ecuación 6.3) por lo que si aplicamos la regla de la cadena para derivar las ecuaciones 6.4 y 6.5 obtenemos:

$$\frac{\partial}{\partial w_0} \text{Perdida}(w) = -2(y - h_w(x)) \quad (6.12)$$

$$\frac{\partial}{\partial w_1} \text{Perdida}(w) = -2(y - h_w(x)) \times x \quad (6.13)$$

La función que deberemos aplicar en cada iteración de este algoritmo es la siguiente:

$$w_i = w_i - \alpha \frac{\partial}{\partial w_i} \text{Perdida}(w) \quad (6.14)$$

Dónde α representa al learning rate.

Si insertamos las ecuaciones 6.6 y 6.7 en la ecuación 6.12, y aplicamos la sumatoria de derivadas, obtenemos las siguientes ecuaciones que nos permitirán minimizar

las pérdidas para las N muestras del dataset de entrenamiento:

$$w_0 = w_0 + \alpha \sum_j (y_j - h_W(x_j)) \quad (6.15)$$

$$w_1 = w_1 + \alpha \sum_j (y_i - h_W(x_j)) \times x_j \quad (6.16)$$

Dado que la superficie de la función de pérdida es convexa no encontraremos mínimos locales, y la convergencia al mínimo global está garantizada. Aún así puede existir un inconveniente: Debemos sumar las N muestras para cada paso o step, el proceso tardará más tiempo mientras mayor sea el espacio de muestras. Un step que cubre todas las muestras de entrenamiento se denomina época o epoch.

6.2.3.1. Gradiente descendente estocástico (Gradiente descendente Online)

Esta variante selecciona aleatoriamente un pequeño número de muestras de entrenamiento en cada step. El efecto inmediato es la reducción de la capacidad de cómputo requerida en cada step, y como el error inherente es proporcional a la cantidad de muestras, el aumento de este es mínimo. Sin embargo, por estas nuevas características, la convergencia no está garantizada, el cálculo puede oscilar alrededor del mínimo sin converger definitivamente. Esta variante también se conoce como Gradiente Descendente Online y puede ser útil entornos en linea o de respuesta inmediata, dónde nuevos datos llegan uno a la vez y las primeras deducciones o descubrimientos no se mantienen en el tiempo. Con un buen modelo seleccionado de base, evoluciona lentamente recordando algo de lo que aprendió en el pasado, pero también adaptándose a los cambios producidos por los nuevos datos.

6.2.4. Naive Bayes

Naive Bayes es un algoritmo de aprendizaje supervisado, que se basa en el teorema de Bayes y se utiliza principalmente en la clasificación de texto que incluye un conjunto de datos de entrenamiento de alta dimensión. Se caracteriza por ser simple y efectivo, y ayuda a construir modelos de aprendizaje automático con el objetivo de hacer predicciones rápidas.

Es un clasificador probabilístico, lo que significa que predice sobre la base de la probabilidad de un objeto. Se basan en la probabilidad condicional.

6.2.5. Probabilidad condicional

La probabilidad condicional se calcula partiendo de dos sucesos o eventos (A y B) en un espacio probabilístico, indicando la probabilidad de que ocurra A dado que ha ocurrido B. Se escribe P (A/B), leyéndose como “probabilidad de A dado B”.

En base a esto, para calcular la probabilidad condicional del evento A dado el

evento B podemos utilizar la siguiente formula:

$$P(A|B) = \frac{P(A|B)}{P(B)} \quad (6.17)$$

6.2.6. Teorema de Bayes

El teorema de Bayes es una proposición que se emplea para calcular la probabilidad condicional de un suceso y fue desarrollado por el matemático y teólogo británico Thomas Bayes. El principal objetivo de este teorema es determinar la probabilidad que posee un suceso comparada con la probabilidad de otro suceso similar. La figura 6.2 representa a alto nivel el flujo de trabajo bajo este modelo.

Hay dos ideas centrales que hacen que un método sea Bayesiano:

- Toda cantidad desconocida es modelada utilizando una distribución de probabilidad de algún tipo.
- Este teorema se usa para actualizar dicha distribución a la luz de los datos.

El teorema es una consecuencia directa de la regla del producto, veamos.

$$p(\theta, y) = p(\theta | y)p(y) \quad (6.18)$$

$$p(\theta, y) = p(y | \theta)p(\theta) \quad (6.19)$$

Dado que los dos términos a la derecha de la igualdad son iguales entre si podemos escribir que:

$$p(\theta | y)p(y) = p(y | \theta)p(\theta) \quad (6.20)$$

Reordenando llegamos al Teorema de Bayes.

$$p(\theta | y) = \frac{p(y | \theta)p(\theta)}{p(y)} \quad (6.21)$$

El cual también suele ser escrito de la siguiente forma:

$$p(\theta | y) = \frac{p(y | \theta) * p(\theta)}{\int_{\theta} p(y | \theta) * p(\theta) d\theta} \quad (6.22)$$

El denominador es una integral sobre todos los valores de θ definidos por el a priori. La integral es reemplazada por una sumatoria en el caso que estemos hablando de valores discretos y no continuos.

Cada término del teorema de Bayes tiene un nombre específico:

$$p(\theta | y) = \text{aposteriori} \quad (6.23)$$

$$p(y | \theta) = \text{likelihood}(\text{verosimilitud}) \quad (6.24)$$

$$p(\theta) = \text{apriori} \quad (6.25)$$

$$p(y) = \text{likelihoodmarginal} \quad (6.26)$$

6.23 A posteriori es la distribución de probabilidad para los parámetros. Es la consecuencia lógica de haber usado un conjunto de datos, un likelihood y un a priori. Se lo suele pensar como la versión actualizada del a priori. De hecho un a posteriori puede ser un a priori de un análisis a futuro.

6.24 Likelihood es la forma de incluir nuestros datos en el análisis. Es una expresión matemática que especifica la plausibilidad de los datos. Es central tanto en estadística Bayesiana como en estadística no-Bayesiana. A medida que la cantidad de datos aumenta el likelihood tiene cada vez más peso en los resultados, esto explica el porqué a veces los resultados de la estadística Bayesiana y frequentista coinciden cuando la muestra es grande.

6.25 A priori es la forma de introducir conocimiento previo sobre los valores que pueden tomar los parámetros. A veces cuando no sabemos demasiado se suelen usar a prioris que asignan igual probabilidad a todos los valores de los parámetros, otras veces se puede elegir a prioris que restrinjan los valores de los parámetros a rangos razonables, esto se conoce como regularización, por ejemplo solo valores positivos. Muchas veces contamos con información mucho más precisa como medidas experimentales previas o límites impuesto por alguna teoría.

6.26 La likelihood marginal (también llamado evidencia) es la probabilidad de observar los datos y promediado sobre todas las posibles hipótesis (o conjunto de parámetros) θ . En general, la evidencia puede ser vista como una simple constante de normalización que en la mayoría de los problemas prácticos puede (y suele) omitirse sin perdida de generalidad. Por lo que el teorema de Bayes suele aparecer escrito como:

$$p(\theta | y) \propto p(y | \theta)p(\theta) \quad (6.27)$$

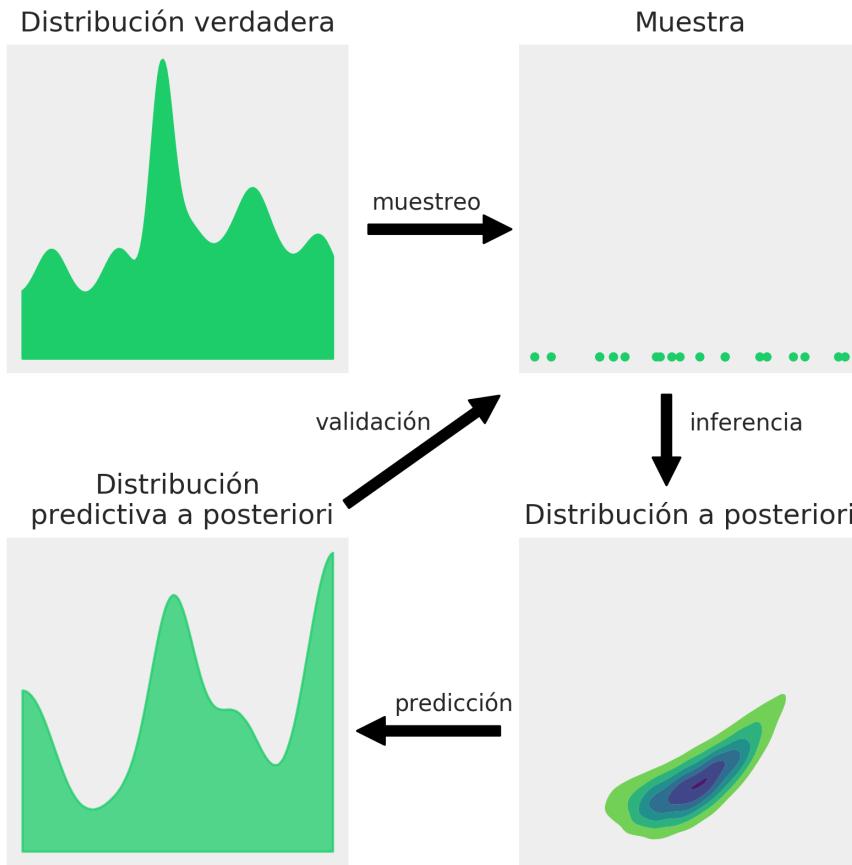


Figura 6.2: Flujo de trabajo bayesiano, imagen traducida por equipo, extraída de [16].

6.2.7. Árboles de decisión

Un árbol de decisión 6.3 es un modelo de predicción que dado un conjunto de datos, fabrica diagramas de construcciones lógicas similares a los sistemas de predicción basados en reglas. Sirven para representar y categorizar condiciones que ocurren de forma sucesiva.

Están formados por nodos vectores de números, flechas y etiquetas:

- **Nodo**: define el momento de toma de decisión entre las distintas opciones, a medida que aumentan los nodos, aumenta el numero de posibles finales.
- **Vectores de números**: solución a la que llega una función entre las distintas posibilidades.
- **Flechas**: la unión entre nodos que representa cada acción.
- **Etiqueta**: define el nombre de cada acción en todos los nodos y flechas del diagrama.

Al mencionar árboles de decisión se debe considerar que cada decisión tiene un costo que se define como dos conceptos distintos, la medición para determinar el valor de

determinada propiedad (atributo), y como el costo de obtener un falso positivo, es decir, decidir que el objeto pertenece a la clase X siendo de la clase Y.

Un concepto importante es el de Entropía, medida que se utiliza para calcular la homogeneidad de las muestras en cada nodo. En los árboles de decisión se busca que tengan entropía más pequeña en sus nodos.

Siguiendo con los conceptos, el índice de Gini es una métrica que mide la frecuencia con la que un elemento al azar sería identificado incorrectamente, al igual que la entropía, se busca un índice de Gini más bajo.

Otro concepto importante es el overfitting (sobreajuste), ya visto anteriormente. Se produce cuando los datos de entrenamiento son pocos o contienen incoherencias y el modelo se sobreentrena. Esto hace que el algoritmo de aprendizaje quede ajustado a unas características muy específicas de los datos de entrenamiento y no tiene relación de causalidad con la función objetivo. En resumidas palabras, el modelo “recuerda” en lugar de aprender a notar características.

El siguiente concepto, Pruning (poda), consiste en eliminar (podar) una rama de un nodo transformando el nodo en una hoja terminal.

Por último es necesario hacer referencia a la validación cruzada que básicamente es construir el árbol con la mayoría de los datos del set de datos considerado, y validarla con la parte menor restante.

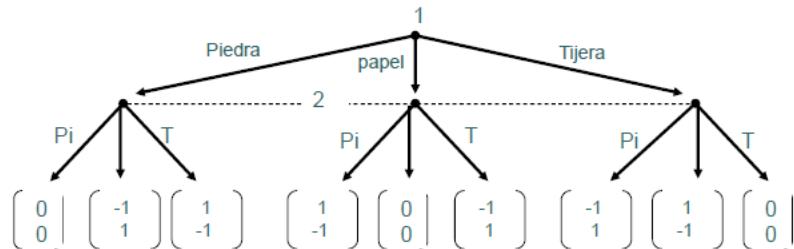


Figura 6.3: Árbol de decisión del juego 'Piedra, papel o tijera' [17].

6.3. Resumen

En este capítulo se hizo un repaso teórico sobre las herramientas que la Inteligencia Artificial nos provee para el desarrollo de aplicaciones. Se comienza explicando la importancia y popularidad que la Inteligencia Artificial ha cobrado en años recientes. Esto da paso a las siguientes secciones del capítulo, en las que se explora con una profundidad media las herramientas de Machine Learning más importantes de las que disponemos haciendo foco sobre las herramientas que se consideraron a lo largo de este desarrollo. El detalle que aquí se expone permite comprender la complejidad de las soluciones propuestas. En próximos capítulos podrá verse su aplicación real y el análisis de los resultados obtenidos.

Capítulo 7

Arquitectura propuesta

7.1. Introducción

Este capítulo detalla los pormenores de la solución implementada. En primera instancia, en la sección 7.2, se mencionará la arquitectura global, diagramando la interacción entre el hardware utilizado y el software implementado.

A continuación, en la sección 7.3 se procede a listar y detallar las distintas soluciones de software periféricas implementadas utilizadas para pruebas y visualización de datos durante la investigación.

Luego, en la sección 7.4 se explica las técnicas utilizadas al momento de preprocesar el set de datos, y de que forma se aplicó dentro del proyecto.

Finalmente en la sección 7.5 se indagará en lo concerniente a técnicas de Machine Learning utilizadas y la metodología de trabajo a la hora de generar las alternativas de modelos.

A lo largo del capítulo nos referiremos al dispositivo de lectura EEG como “casco EEG”.

7.2. Arquitectura de solución propuesta

La solución final desarrollada 7.1 consta de tres aplicaciones. Primero y principal la aplicación “Conector Node.js para lectura de datos reales”. Esta aplicación es la que posibilita la comunicación de la computadora con el casco EEG para obtener lecturas en vivo, y respaldo de esas lecturas en la nube. En segundo lugar, el sitio web “Gráfico en vivo” que permite ver sobre un gráfico temporal las lecturas en crudo que se obtienen desde el casco EEG, con actualizaciones en vivo en el momento. En tercer lugar, la aplicación “Laberinto”, que se utilizó como principal experimento luego de lograr la traducción de las lecturas del casco EEG en instrucciones.

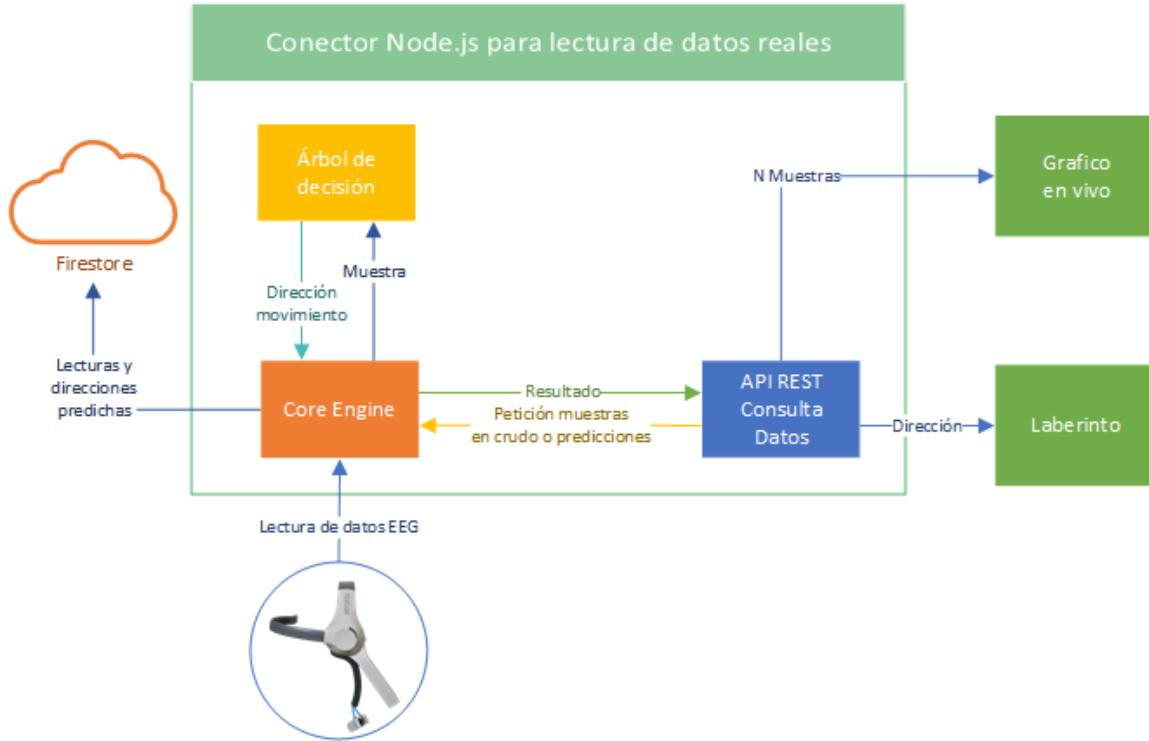


Figura 7.1: Diagrama de interacción entre las distintas partes de la solución

7.3. Soporte a la investigación

Con el fin de hacer pruebas de funcionamiento y visualización de datos, se optó por desarrollar una serie de aplicaciones sencillas de soporte. Todas las aplicaciones se desarrollaron sobre Node.js, y en los casos de las aplicaciones web, se utilizó el framework Vue.js para la construcción a nivel visual y funcionamiento en el frontend. A continuación se presentan las aplicaciones desarrolladas con detalles de implementación.

7.3.1. Conector Node.js para lectura de datos reales

Este conector es utilizado como nexo entre el casco EEG y el aplicativo final. El desarrollo se basa en 2 módulos claramente identificables, por un lado el Core Engine, que permite la conexión con el casco EEG, y por otro lado una API REST como mecanismo de consulta que implementa los métodos por los cuales se accede a la información. El desarrollo implementa las librerías Morgan, net, events, utils y Express.js.

7.3.1.1. Core Engine

El Core Engine es una implementación del protocolo TGSP especificado por el fabricante que permite acceder a las lecturas del casco EEG. Dicho protocolo está basado en JSON para la transmisión y recepción de datos de ondas cerebrales ThinkGear de manera estándar entre un cliente y un servidor.

A continuación se detalla técnicamente el estándar de datos (peso, tipo y representación) así como también la forma de transmisión y velocidad para el casco EEG, adoptado por el equipo para llevar a cabo las distintas pruebas Mindware MW001 1.5v a 95 mA.

Por defecto, el protocolo de comunicación Thinkgear utilizado por el casco EEG emite un evento 8BIT_RAW Wave Value que permite manejar valores de 1 Byte sin signo, equivalente a valores de onda RAW con signo, con la diferencia de estar escalado para no contar con signo y emite solo los 8 bits (MSB) más significativos definidos por el HW. Esto hace posible generar valores de onda sin procesar, teniendo en consideración las restricciones de ancho de banda de las comunicaciones en serie a una velocidad de 9600 baudios, a costo de no generar el par de bits (LSB) de valores menos significativos. Para muchas aplicaciones (Por ejemplo, visualización de gráficos de ondas cerebrales en tiempo real), una precisión de 8 bits es más que aceptable teniendo en cuenta que el ojo humano normalmente no puede discernir rápidamente píxeles que pueden corresponder a bits de precisión LSB.

Internamente, el casco EEG realiza los cálculos en función a la máxima precisión de la información de onda sin procesar disponible, por lo que en sí no se descarta ninguna información internamente, solo el valor bruto generado y emitido desde el casco EEG se reduce a 8 bits para ahorrar ancho de banda en serie.

En caso de precisar más definición en la salida, se puede setear 16BIT_RAW Wave Value como valor predeterminado y cambiar la velocidad de transmisión a 57600 baudios debido a la extensión y peso de los datos enviados. Para el desarrollo se trabajó con el modo por defecto.

En ambos casos los valores se captan 128 veces por segundo o aproximadamente 1 valor cada 7.8ms, y son emitidos de manera asincrónica. Generalmente en los sistemas se utiliza el estándar de obtención de 1 valor/segundo.

Para los valores de ondas cerebrales los cascós con módulo TGAT, cómo es nuestro caso, las bandas de frecuencias consideran 8 números de punto flotante de 4 bytes en el siguiente orden:

- delta (0.5 - 2.75Hz)
- theta (3.5 - 6.75Hz)
- low-alpha (7.5 - 9.25Hz)
- high-alpha (10 - 11.75Hz)
- beta baja (13 - 16.75Hz)
- beta alta (18 - 29.75Hz)
- gama baja (31 - 39.75Hz)
- gama media (41 - 49.75Hz)

Estos valores no tienen unidades y, por lo tanto, solo son significativos cuando se comparan entre sí y consigo mismos, para considerar la cantidad relativa y las

fluctuaciones temporales. El formato de punto flotante es el estándar big-endian de IEEE 754, por lo que los 32 bytes de los valores se pueden convertir directamente como un flotante * (en entornos big-endian) para usarse como una matriz de flotantes.

El casco comunica sus señales a un Dongle USB RF Mindwave MW002 5v a 60 mA (receptor de radio frecuencia por USB) que toma un puerto COM (cuyo número depende del puerto al cual conectemos, variando entre 3 y 5 dependiendo de la cantidad de puertos de la computadora). El mismo crea un servidor local con Host “localhost” y puerto predefinido 13854.

El flujo en serie debe analizarse e interpretarse como paquetes ThinkGear conformados de la siguiente manera:

1. Packet Header
2. Packet Payload
3. Payload Checksum

Cada paquete se envía en el orden mencionado, un paquete válido ocupa un mínimo de 4 bytes (solo en caso de que los valores en payload sean 0 o vacíos), y un máximo de 173 bytes (en caso de que los valores de payload estén completos ocupando 169 bytes).

El estándar implementado aporta robustez y flexibilidad a través del encabezado y los checksum, que proporcionan verificación de la integridad de los datos y la sincronización del flujo de datos, mientras que el formato de payload permite determinar que campos de datos se desea agregar/quitar de la tira de datos provistos, esto garantiza que en caso de cambiar de casco por uno que provea menor o mayor cantidad de datos, no será necesario cambiar el protocolo en absoluto.

Código fuente 7.1: Ejemplo de Paquete ThinkGear, tomado de [18]

```
1 0xAA // [SYNC]
2 0xAA // [SYNC]
3 0x08 // [PLENGTH] (payload length) of 8 bytes
4 0x02 // [CODE] POOR_SIGNAL Quality
5 0x20 // Some poor signal detected (32/255)
6 0x01 // [CODE] BATTERY Level
7 0x7E // Almost full 3V of battery (126/127)
8 0x04 // [CODE] ATTENTION eSense
9 0x12 // eSense Attention level of 18%
10 0x05 // [CODE] MEDITATION eSense
11 0x60 // eSense Meditation level of 96%
12 0xE3 // [CHKSUM] (1's c inverse of 8-bit Payload sum of 0x1C)
```

Estos paquetes de transmisión asincrónica pueden ser recibidos a través de un puerto serie COM o USB y consumidos por cualquier lenguaje o plataforma que sea capaz de interpretarlos.

La investigación, se centrará en los valores provistos por la estructura de payload que básicamente son los valores no procesados provistos por el casco donde se puede destacar el estado de atención, meditación, el set de ondas cerebrales, y la calidad de señal. La interpretación del protocolo se realizará sobre Node.js creando una librería que internamente implementa un event listener que constantemente escucha el puerto y host mencionados anteriormente, logrando captar cada modificación que surja sobre esta dirección. Al momento de lanzar la librería se pueden configurar algunos parámetros, tales como host, puerto o rawOutput.

Como estándar de trabajo y por practicidad, el resultado de una consulta retornará una estructura en formato JSON con los siguientes datos:

Código fuente 7.2: Ejemplo de Paquete ThinkGear

```
1 {
2     "data": {
3         "ts": datetime (generado al tomar la medición)
4     },
5     "eSense": {
6         "attention": %,
7         "meditation": %
8     },
9     "eegPower": {
10        "delta": val,
11        "theta": val,
12        "lowAlpha": val,
13        "hiAlpha": val,
14        "lowBeta": val,
15        "hiBeta": val,
16        "lowGamma": val,
17        "hiGamma": val,
18    },
19    "poorSignalLevel": val
20 }
```

A la misma se agregará la fecha y hora de la medición para poder representar la variación de las distintas ondas cerebrales en el tiempo.

7.3.1.2. Escritura en Firestore

En un primer momento, para persistir los datos se utilizó un archivo JSON general, que permitía almacenar la información en tiempo real que iba surgiendo de las mediciones del casco. Esta alternativa brindaba:

- Poca flexibilidad: al momento de compartir dichos datos ya que su almacenamiento dependía netamente de la máquina mediante la cual se tomaran las mediciones
- Escasa seguridad del archivo: dependiendo completamente de la seguridad de la computadora en donde quedara alojado

- Disponibilidad acotada: para servir la información a múltiples clientes generaba dependencia total de la computadora que lo alojara

Debido a la gran cantidad de inconvenientes de esta implementación se optó por agregar una capa mas para la persistencia de los datos. Para llevar adelante esta implementación de la persistencia de datos, se optó por utilizar Firestore, la base de datos de Firebird que permite:

- Guardar datos como una colección de documentos con una estructura muy similar a un archivo JSON
- Admite el almacenamiento de datos locales para las apps que funcionan sin conexión
- Recupera, ordena y filtra datos mediante consultas.
- Las consultas son superficiales: solo muestran documentos de colecciones o grupos de colecciones específicos y no muestran datos de subcolecciones
- Las consultas se indexan de forma predeterminada: el rendimiento de las consultas es proporcional al tamaño del conjunto de resultados, no del conjunto de datos, este punto es vital para el procesamiento de datos en nuestro sistema, ya que es crucial que se puede acceder y servir los datos en el menor tiempo posible.

7.3.1.3. Diagramar toma y escritura de datos Firestore.

Si bien una de las cualidades de Firestore es poder almacenar datos localmente, ese mecanismo solo lo utilizaríamos en caso de no disponer de conexión a internet, el desarrollo apunta a persistir la información en la nube, y los clientes autorizados, puedan consumirla en tiempo real, para esto se desarrolló un modulo que habiendo prendido el casco, permite activar la lectura de señales, tomar la información provista por el casco mediante el Core Engine mencionado en la sección 7.3.1.1, estructurar la información acorde a la necesidad en una colección que denominamos “freq”, con un documento por cada frecuencia recibida del casco identificado con la fecha y hora exacta de su adquisición, y una estructura similar a la que se había ideado en el archivo JSON local como se aprecia en 7.2.

tesina-329921	freq	2021-12-13T00:35:50.657Z	
+ Iniciar colección	+ Agregar documento	+ Iniciar colección	
freq >	2021-12-13T00:35:39.700Z 2021-12-13T00:35:40.209Z 2021-12-13T00:35:40.721Z 2021-12-13T00:35:41.229Z 2021-12-13T00:35:41.740Z 2021-12-13T00:35:42.263Z 2021-12-13T00:35:42.768Z 2021-12-13T00:35:42.978Z 2021-12-13T00:35:43.852Z 2021-12-13T00:35:44.709Z 2021-12-13T00:35:45.686Z 2021-12-13T00:35:46.692Z 2021-12-13T00:35:47.677Z 2021-12-13T00:35:48.666Z	eSense attention: 47 meditation: 56 eegPower delta: 3449 highAlpha: 651 highBeta: 1259 highGamma: 6374 lowAlpha: 3812 lowBeta: 2571 lowGamma: 545 theta: 4198 oorSignalLevel: 0	

Figura 7.2: Captura de la estructura modelo propuesta para persistencia de datos en Firestore, en funcionamiento

7.3.1.4. API REST para consulta de datos de lectura

Los datos del casco pueden ser consumidos a través de una API REST por 4 factores fundamentales para el desarrollo.

- Escalabilidad: Gracias a la separación entre cliente servidor, toda la lógica del servidor provisto por la API REST es transparente al cliente y escalar su funcionamiento no implica grandes esfuerzos.
- Flexibilidad y portabilidad: Requisito imprescindible de que los datos sean enviados de manera correcta independientemente de donde esté alojado el servicio.
- Independencia: Separación de lógica entre cliente y servidor, facilitando el consumo de la misma a distintas plataformas de trabajo y entornos de desarrollo brindando facilidad de uso, y respuestas estandarizadas.

El servicio implementa Express v4.17.1, para facilitar el servicio de enrutamientos, y Morgan v1.10.0 para las definiciones necesarias de middleware y entorno. El servicio tiene comunicación directa con la base de datos FireStore mencionada en la sección anterior que posibilita tener acceso a los datos del casco de manera directa, sin intermediarios.

Al momento de ejecutar el servidor de Node.js que provee la funcionalidad, se implementa la librería nodemon v2.0.7, que nos permite determinar la ruta de ejecución y el tipo de entorno a ejecutar, dev (Desarrollo) o prod (Productivo).

Actualmente el servicio de enrutamiento cuenta con 3 rutas:

- http://url_servidor:port/single devuelve la lectura más reciente del casco.

- `http://url_servidor:port/all` devuelve todos los datos censados por el casco una vez conectado el mismo.
- `http://url_servidor:port/getDirection` devuelve una predicción de dirección basada en la lectura eeg mas reciente.

Donde `http://url_servidor` hace referencia a la ip o dominio sobre el cual se está ejecutando la instancia del servidor que ejecuta el core engine y la api de consulta. En tanto port hace referencia al puerto sobre el cual se realiza la consulta, ejemplo 80.

Con esto se brindará una funcionalidad que se adapte a la necesidad del cliente en distintos momentos, por ejemplo, para comenzar una visualización de datos en un gráfico de tiempo real, inicialmente se podría solicitar la lista de valores completa, y en cada actualización que sufra la visualización solo pedir el último valor, y agregarlo manualmente al set de valores considerado con anterioridad, considerando que se manejan volúmenes de datos realmente grandes, este mecanismo de consulta de datos brindará una mejor performance.

7.3.2. Gráfico en vivo

Es una aplicación web que permite visualizar en vivo mediciones EEG obtenidas a través del casco. Fue implementada considerando una estructura común a todo el proyecto, para la manipulación de los registros de mediciones obtenidas mediante el casco. Esta estructura común consiste en objetos de tipo JSON. Considerando que el tipo de ondas cerebrales utilizadas para el proyecto es el tipo Alfa, esta aplicación utiliza sólo esta información de la disponible en cada registro recibido en formato JSON.

7.3.2.1. Funcionamiento

Tan pronto cómo es accedida, la aplicación comienza hacer peticiones de información a una dirección URL configurada. Estas peticiones se hacen de forma periódica, manteniendo siempre una cantidad fija de muestras en pantalla y refrescando el gráfico para descartar las muestras más antiguas. Tanto el tiempo de intervalo entre peticiones, como la cantidad total de muestras visibles en pantalla son valores configurables, cuyos cambios son aplicados de forma inmediata.

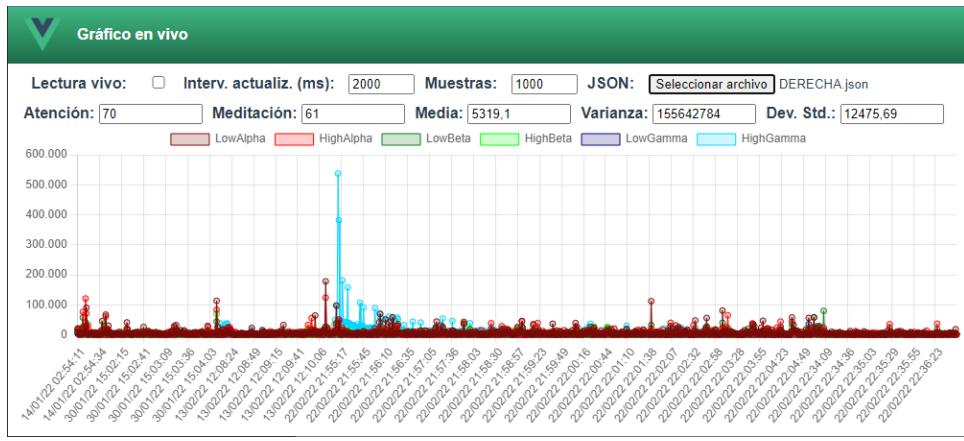


Figura 7.3: Captura de la aplicación “Gráfico en vivo”, en funcionamiento

7.3.3. Laberinto

Para probar el proceso de traducción de ondas cerebrales en instrucciones de dirección, se decidió implementar una aplicación básica que implique algún tipo de movimiento dirigido, para lograr representar de forma gráfica las instrucciones enviadas por el usuario.

Para ahorrar tiempos de desarrollo, se optó por reutilizar parte de un proyecto personal de práctica, implementado mediante Angular, un framework similar al que se utiliza para los desarrollos web de esta tesina, Vue.js. El mencionado proyecto contaba con una implementación para la lógica básica de movimiento dentro de un tablero representado por una estructura de matriz. Consideramos este tipo de representación y su sencillez, el entorno ideal para las pruebas de captura de instrucciones, donde sería rápido ver el efecto de las decisiones del usuario, reconocer posibles errores y aplicar correcciones necesarias. La migración entre ambos frameworks resultó un proceso sencillo, dado que tanto Angular como Vue.js se basan en la noción de componentes, es decir que estructuralmente ambas aplicaciones son similares. En primera instancia se simplificó el juego, llegando así al modelo de laberinto, un juego en el cual solo se necesita mover un puntero desde un punto A a un punto B, moviéndose en cuatro direcciones para avanzar.

En segundo lugar fue necesario aislar la lógica de movimientos de los eventos de teclado, dado que el impulso de movimiento en este caso es recibido desde otro tipo de fuente, así la acción es independiente del evento que la ocasiona, y se puede invocar desde cualquier método.

Finalmente se implementó un EventListener que se encuentra escuchando permanentemente a la espera de un impulso enviado por el usuario para efectuar un movimiento.

Este EventListener deberá comunicarse con el Conejero Node.js, para obtener instrucciones generadas a partir de las lecturas obtenidas del casco de lectura.

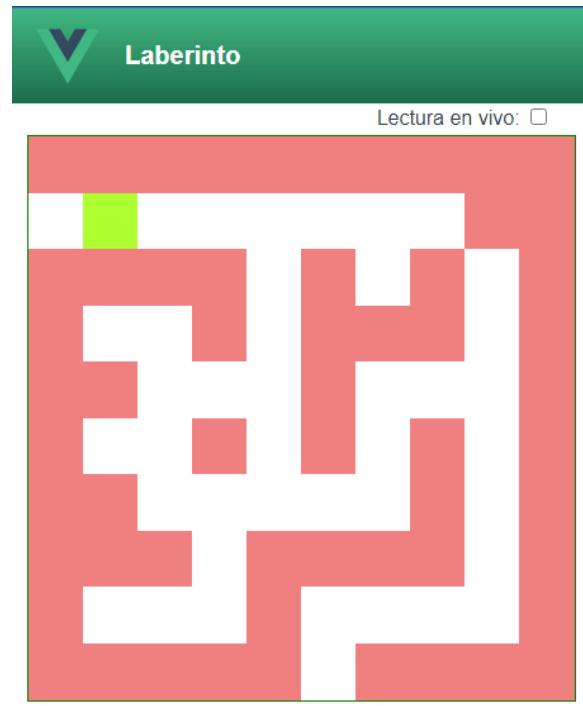


Figura 7.4: Captura de la aplicación “Laberinto”

7.4. Preparación del set de datos

El preprocesamiento de datos es una etapa esencial del proceso de descubrimiento de información o KDD (Knowledge Discovery in Databases, en inglés). Ésta etapa se encarga de la limpieza, integración, transformación y reducción de los datos, lo cual es necesario ya que por lo general el uso de datos de baja calidad implica un proceso de machine learning con resultados muy pobres o nulos.

En sistemas BCI, se han propuesto y aplicado muchas técnicas para la interpretación de señales, tales como: análisis en el dominio del tiempo y frecuencia, filtros adaptados, potencia espectral, Patrones Espaciales Comunes (CSP), Transformada Rápida de Fourier (FFT), Modelos Autorregresivos (AR), Transformada Discreta de Wavelet (DWT), Transformada Discreta Wavelet Packet (DWPT), Entropía, Transformada de Hilbert, Redes Neuronales y difusas, y combinaciones entre ellas.

Estas técnicas se pueden dividir en tres grupos principales, que son:

- Los métodos que sustraen información temporal de la señal.
- Métodos que extraen información frecuencial.
- Métodos híbridos, basados en representaciones tiempo-frecuencia, que aprovechan tanto la información temporal y frecuencial.

Después de aplicar la fase de preprocesamiento, el conjunto resultante puede ser visto como una fuente consistente y adecuada de datos de calidad para la aplicación de algoritmos de minería de datos. El preprocesamiento incluye un rango amplio de

técnicas que podemos agrupar en dos áreas: preparación de datos y reducción de datos.

La preparación de datos tiene como objetivo inicializar correctamente los datos que servirán de entrada para nuestro modelo de aprendizaje mediante una serie de técnicas consideradas como de uso “obligado”, ya que sin ellas los algoritmos de extracción de conocimiento no podrían ejecutarse u ofrecerían resultados erróneos.

Por otro lado, las técnicas de reducción de datos se orientan a obtener una representación reducida de los datos originales, manteniendo la mayor medida posible de la información e integridad existente, es por ello que estas técnicas de reducción no son estrictamente necesarias. En éste área, la técnica más relevante es la selección de atributos.

7.4.1. Transformada rápida de Fourier (FFT)

La transformada de Fourier rápida (FFT) es una potente herramienta de análisis matemático que se utiliza en diversos campos de la ciencia. Se utiliza para indagar la distribución de la amplitud de espectro en EEG y extraer el pico del espectro para reflejar las diferentes tareas del cerebro. Mediante el uso de la FFT, la señal de EEG puede mapearse desde el dominio del tiempo al dominio de la frecuencia. El espectro de frecuencia de la señal es reconocido por descomponer la señal en su correspondiente sinusoidal de diferentes frecuencias. Para el presente trabajo la utilizamos concretamente porque es capaz de trabajar con multiplicación de números grandes y el cálculo aproximado de frecuencias de señales analógicas unidimensionales sirviendo particularmente para la extracción de características en la clasificación de tareas motoras.

La FFT puede ser calculada de forma relativamente rápida casi en tiempo real, por lo que se ajusta bastante a la necesidad del desarrollo propuesto.

7.5. Modelos de Machine Learning

El machine Learning o aprendizaje automático consiste básicamente en automatizar, mediante distintos algoritmos, la identificación de patrones o tendencias que se “esconden” en los datos. Por ello, resulta muy importante no sólo la elección del algoritmo más adecuado (y su posterior parametrización para cada problemática concreta), sino también el hecho de disponer de un gran volumen de datos de suficiente calidad.

Por el tipo de problema a analizar, y desconocimiento de una solución a priori, nos centramos en la comparación de 3 modelos distintos.

- Regresión Lineal simple 6.2.2.1
- Naive Bayes 6.2.4
- Árbol de decisión 6.2.7

Para utilizar los modelos referenciados anteriormente, se hizo uso del framework TensorFlowJS. En algunos casos (por ejemplo en el caso del árbol de decisión), se debió utilizar una librería de terceros nodejs-decision-tree-id3¹ que básicamente es una implementación de arboles de desición en node.js mediante la utilización del algoritmo ID3.²

7.6. Como entender resultados

Para el análisis y entendimiento de los datos resultantes se optó por utilizar una matriz de confusión.

Una matriz de confusión o matriz de error, es una tabla de contingencia que sirve como herramienta estadística para el análisis de observaciones emparejadas. Como indican Comber et al. (2012), la matriz de confusión ha sido adoptada, de facto y de jure, como un estándar para informar sobre la exactitud temática de cualquier producto de datos derivados de la teledetección. En esta línea, la matriz de confusión aparece reconocida en la Norma Internacional ISO 19157. [19]

Básicamente es una tabla matriz cuadrada de dimensión MxM (filas x columnas), donde M indica las clases a ser evaluadas, y los resultados correctos se ubican en la diagonal principal.

Clase	R1	R2	...	RM	
Conjunto	G1	N11	N21	...	N2M
De	G2	N21	N22	...	N2M
Datos
	GM	NM1	NM2	...	NMM
Total		N+1	N+2	...	N+M

Cuadro 7.1: Notación para matriz de confusión

¹<https://github.com/serendipitous/nodejs-decision-tree-id3>

²https://en.wikipedia.org/wiki/ID3_algorithm

7.7. Resumen

En este capítulo se hizo foco sobre las herramientas utilizadas y desarrolladas, describiendo en primer lugar la arquitectura propuesta para concluir con las distintas alternativas de clasificación que se consideraron.

Comienza explicando como se ideó la solución, la arquitectura 7.2 y como interactúan los distintos componentes dando lugar a la siguiente sección en donde explicamos las herramientas que creamos para dar soporte a la investigación 7.3.

En la siguiente sección se explica sobre el tratamiento de los datos 7.4, para finalmente concluir con los modelos considerados 7.5 para resolver el problema planteado.

El próximo capítulo está únicamente enfocado en su aplicación real y el análisis de los resultados obtenidos.

Capítulo 8

Resultados y discusión

8.1. Introducción

En este capítulo se documenta los resultados de la solución implementada. Para ello se explica el origen del set de datos utilizados y los ajustes realizados a los valores obtenidos, así como también los métodos y librerías elegidas, la configuración del modelo de machine learning empleado, y finalmente el análisis de los experimentos realizados y sus resultados.

8.2. Obtención de datos

La tarea de obtener datos del casco a partir de alguna acción física realizada no es sencilla, ya que demanda bastante tiempo.

Para el desarrollo, se preparó un set de datos primarios con el casco puesto tratando de imaginar el movimiento deseado. Determinamos 4 movimientos básicos:

1. Arriba
2. Abajo
3. Derecha
4. Izquierda

Para lograrlo, la persona que esté tomando la muestra se concentra en un punto fijo e imagina la acción durante un periodo de tiempo definido 15 minutos por cada acción, teniendo como objetivo generar 1000 muestras por cada movimiento. La periodicidad de adquisición de datos es aproximadamente 1 muestra por segundo, en algunos casos más.

El momento en que toma las muestras se realiza en un ambiente sin ruidos externos, sin distracciones, confiando en que, el enfoque que se pudiera lograr fuera el mejor posible para ejecutar la acción.

Antes de seguir describiendo el proceso, es bueno aclarar que en función de la edad de la persona existen patrones normales de ondas cerebrales, pero no necesariamente 2 personas de igual edad emiten el mismo patrón de frecuencias al ejecutar una acción.

También está comprobado científicamente que el ritmo de las ondas cerebrales disminuye ligeramente entre 7-8 hz conforme la persona va avanzando en edad, por lo que es recomendable que se ajuste el modelo en personas mayores a 65 años, ver artículo [20].

8.3. Datos primarios

Al set de datos obtenidos a partir de la información tomada del casco sin ningún tipo de procesamiento fue definido como datos primarios, o datos sin procesar. Este set de valores es la fuente de entrada para el motor de pre-procesamiento de datos.

La calidad de los mismos influyen directamente sobre la performance del modelo, el no procesamiento implica un muy bajo rendimiento, poco acierto y mayor desvío.

8.3.1. Calidad datos primarios

El espectro de datos sin procesar obtenidos están expresados en la unidad especificada por el fabricante, definida en la sección 5.2.2.

Este set de datos sin procesar no aporta un nivel significativo de acoplamiento, por lo que es muy difícil tomar una definición con datos tan abstractos y con poca relación. El objetivo principal en este punto fue relevar las características mas importantes del set de datos sin procesar, entenderlos, detectar los puntos muertos y datos erróneos, en base a esto obtener un set de datos mas limpio, útil para el análisis.

8.3.2. Datos erróneos

El set de datos sin procesar proporciona información de una tarea específica, pero también proporciona información errónea, por ejemplo los tiempos muertos donde el cerebro emite frecuencias, pero no son representativas para la acción que se trata de realizar, aparte de proporcionar señales estocásticas que contienen mucho ruido.

8.4. Balanceo de clases

El primer paso para preparar los datos justo antes de entregarlos al clasificador es verificar el balance entre sus clases y, en caso de que este balance no esté

dado, tomar las medidas para que el balance sea el mayor posible: contar con clases balanceadas permite que las métricas de los clasificadores no se inclinen por una u otra clase durante el entrenamiento. Esta inclinación o sesgo en las métricas de clasificación de las muestras suele darse hacia la clase más densa dentro del conjunto de datos.

Durante la adquisición de los datos a través del casco, se fijó un tiempo específico de duración por cada movimiento pensado, pero no se tiene forma de corroborar que los datos del movimiento puntual que se estaba obteniendo fueran en su totalidad correctos, por lo que a pesar de garantizar 15 minutos por movimiento, no se pudo afirmar tener la misma cantidad de muestras por movimiento.

Ante este escenario, se requirió realizar un balanceo de las clases para que el entrenamiento de los clasificadores no se viera permeado por sesgos hacia determinadas clases solo por su densidad dentro del conjunto de datos de entrenamiento. Se optó por fijar un límite de 1000 muestras por movimiento (no necesariamente correctas) y continuar con las siguientes etapas de procesamiento.

8.5. Pre procesamiento de datos

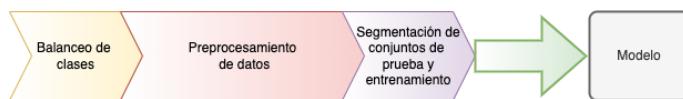


Figura 8.1: Flujo de procesamiento de datos

Para extraer información de señales estocásticas que contienen mucho ruido, se requiere un análisis exhaustivo de cada una y entender como se relacionan entre ellas.

Esta etapa del desarrollo es la que mas tiempo y esfuerzo tomó, básicamente porque obtener un conjunto de datos analizables con el set de datos sin procesar como base fue una tarea realmente compleja. Es un hecho comprobado que la tarea de pre procesamiento de datos es una etapa crucial, que determina en gran medida el éxito de un proyecto de machine learning o minería de datos. Por lo cuál, fue de suma importancia lograr llevar el set de datos a un estado aceptable. Para lograrlo se hizo un proceso de prueba y error con algunas técnicas descartadas y variaciones de las que se utilizaron finalmente. A continuación se presentan los distintos algoritmos y técnicas de procesamiento que llegaron a la versión final:

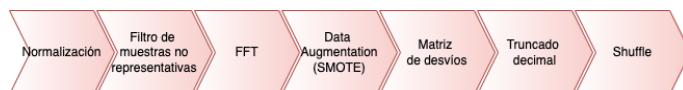


Figura 8.2: Etapa de pre procesamiento de datos

8.5.1. Normalización

Puesto que el set de datos sin procesar obtenido presentaba muestras con gran variabilidad entre las distintas frecuencias, esto resultaba en problemas para el mo-

delo de árbol de decisión que se utilizó en ese momento, ya que generaba una gran cantidad de hojas y no resolvía correctamente. Se presentó la necesidad de normalizar los datos para tener unidades más simples.

Se tomó la decisión que el método que mejor se adaptaba a nuestra necesidad era MinMax Scaler, ya que transforma linealmente los datos originales, obteniendo un set de datos equivalente donde todos los datos están incluidos dentro del intervalo $[0,1]$.

$$xScaler = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (8.1)$$

8.5.2. Filtro de muestras no representativas

Teniendo como premisa que no todos los valores de la muestra podían ser representativos, a partir del set de datos normalizados, se procedió a limitar los valores dentro de un rango considerado representativo, para esto, se partió de la base de que dentro de cada registro, al menos 1 valor aportara información útil.

Luego se procede a agrupar estos registros donde cada valor de cada una de las características se encuentre dentro del rango alcanzado por estos valores.

8.5.3. Transformada rápida de fourier (FFT)

Con el set de datos un poco mas limpio de errores, se convirtieron los valores a frecuencias, porque hasta el momento solo se tenía un set de datos numéricos con poco sentido. El gráfico 8.3 a continuación trata de mostrar de que manera llegan las señales sin procesar.

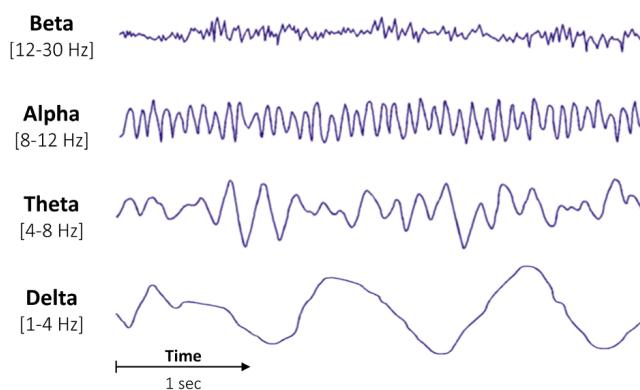


Figura 8.3: Espectro de señal cruda.

Se evaluaron los distintos métodos, y se decidió aplicar la Transformada Rápida de Fourier (FFT), que se utiliza para indagar la distribución de la amplitud de espectro en EEG y extraer el pico del espectro para reflejar las diferentes tareas del cerebro [21]. Mediante el uso de la FFT, la señal de EEG puede mapearse desde el dominio del tiempo al dominio de la frecuencia.

Al dividir la señal sin procesar en bandas de frecuencia, se puede eliminar el ruido (componentes de alta frecuencia) ya que se convirtieron los valores de señales crudas en ondas sinusoidales simples.

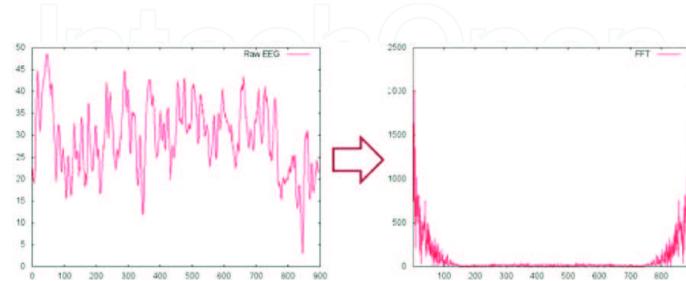


Figura 8.4: Espectro de señal sin procesar, y técnica de transformada rápida de fourier aplicada. Imagen tomada del libro [22]

8.5.4. Data augmentation: SMOTE

Synthetic Minority Oversampling Technique: técnica estadística de sobremuestreo de minorías sintéticas, se utiliza para aumentar el número de casos de un conjunto de datos de forma equilibrada. Busca equilibrar el desbalance de clases.

La implementación de esta técnica dentro de nuestro esquema de pre procesamiento de datos es fundamental, ya que, a pesar de que se logró por cada clase, una cantidad de muestras uniformes, no podemos garantizar que dichas muestras sean fieles 100 % a la clase que tratamos de pensar. Por lo que la técnica de sobremuestreo, nos permitió en término de características, equilibrar los conjuntos pensados. También es importante remarcar que como el proceso de toma de muestras un proceso lento, surge la necesidad de implementar una manera de aumentar los datos de los que se disponía para poder avanzar en la resolución de problemas de código.

8.5.5. Matriz de desvío respecto a media y desvío estándar

A esta altura, se provee un set de datos semi procesados, pero con características que no tienen relación entre si. Para que el modelo pueda interpretar correctamente que acción tomar, tiene que existir algún tipo de relación en los datos que le permita al modelo diferenciarlo claramente.

En función de esto es que se procedió a armar un set de datos en base a los previamente procesados. Para esto se realizaron los siguientes pasos:

1. Se toma el valor promedio y desvío estándar de cada colección completa de frecuencia de la matriz de datos. Es decir, un par (promedio, desvío estándar) por cada frecuencia del set de datos.
2. Se toma el valor promedio y desvío estándar de la muestra en particular.

3. Por cada frecuencia de la muestra se generan 2 valores nuevos, la diferencia entre el valor de la muestra y el valor promedio global, y la diferencia entre el valor de la muestra y el valor de desvío estándar global.
4. Se persiste en una nueva matriz las diferencias tomadas, y el valor promedio y desvío estándar de cada muestra.

Por lo que, de una muestra con 8 variables, ahora se obtiene una nueva muestra de 18 variables (Los 16 valores, obtenidos a partir de los 8 valores originales, más el promedio y desvío estándar de cada muestra). Esto expone más fuertemente el grado de relación de cada muestra con su etiqueta (La dirección de movimiento) y la relación respecto del dataset total, facilitando la comparación de las muestras de nuestra matriz. Esta etapa es crucial y fue el punto de trabajo más fuerte de la etapa de preprocessamiento. Agregándola se logró un crecimiento exponencial y consistente de la precisión del modelo de machine learning que eventualmente seleccionamos.

8.5.6. Truncado de valores

Con el set de datos de pruebas que se realizaron a medida que se avanzaba en el procesamiento de los datos, se encontró que a pesar de tener un sólido conjunto de valores, el modelo no terminaba de clasificar bien.

Por esto, se recreó el modelo completo en Python, y efectuando las mismas pruebas, obteníamos mejores resultados.

Entonces se procedió a identificar que modificaciones/restricciones tenía el lenguaje que afectaban tan claramente al desempeño de nuestro modelo. Se revisó el conjunto de valores obtenidos en la matriz, a partir de la cual se observa que cada nueva frecuencia contaba con una cantidad absurda de decimales, en muchos expresados en notación científica y nos topamos con un bug del lenguaje,

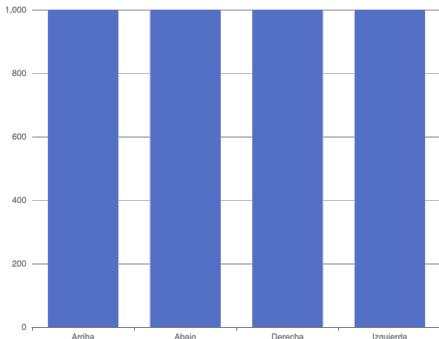
8.5.7. Shuffle

Habiendo obtenido una matriz de datos, a nuestro parecer consistentes, se decidió mezclarlos para no darle al modelo un set de valores y clases correctamente ordenados, esto ayudará a evitar que el árbol tienda a tomar una errática por alguna clase en particular.

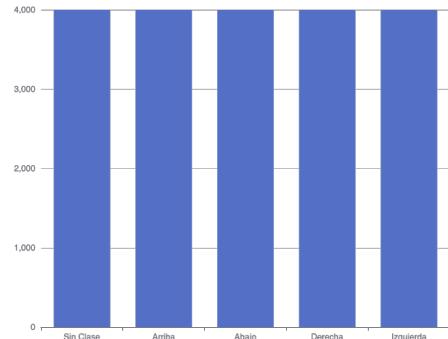
8.6. Segmentación del set de datos

Una vez realizado el balanceo de las clases y el pre procesamiento mediante las técnicas antes mencionadas, se separaron los datos que serían utilizados como datos de entrada para el entrenamiento y validación del modelo construido de los datos utilizados posteriormente en el proceso de prueba del mismo.

Inicialmente se consideró un set de datos de 4000 muestras, 1000 por cada clase. Luego se re-mapeo el conjunto para considerar muestras no representativas. Para finalizar, se balancearon las clases obteniendo un set de datos de 4000 valores por clase, dicho set de datos se utilizó para hacer las comparaciones entre distintos modelos en pos de evaluar su desempeño.



(a) Se de datos original.



(b) Se de datos balanceados considerando la clase no representativa, 4000 muestras por clase.

Figura 8.5: Set de datos original, 4 clases, 1000 valores por clase. Se de datos balanceado posterior a re-mapear las clases, 5 clases, 4000 valores por clase.

A partir de estas muestras, para el modelo definitivo de árbol de decisión se aplica data augmentation llegando a generar un conjunto total de 750.000, se realizó un ordenamiento aleatorio de los datos y se obtuvieron 2 conjuntos, uno para entrenamiento contenido el 70 % de 525.000 muestras de todas las clases; dado que las clases se encuentran balanceadas, la cantidad de muestras seleccionadas para cada clase fue 105.000.

Consecuentemente, el 30 % restante serán utilizados para testear algoritmo de clasificación, constituyendo así 45.000 muestras de prueba para cada clase y un total de 225.000 muestras de prueba para todo el set de datos.

8.7. Modelos de clasificación

En primer término esta etapa pretende servir como marco de comparación con respecto a los distintos modelos evaluados. Todas las pruebas se realizaron en iguales condiciones de uso del casco, con la misma cantidad de datos observados.

El desempeño de los distintos modelos será representado a través de una matriz de confusión. Ésta es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado.

Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo dos clases.

Se tomó este criterio de evaluación porque los resultados generados a partir de los datos de entrada al modelo son muy variables, por lo que la medida del error cuadrático medio puede no ser representativa, ya que el clasificador en todo momento obtiene una clasificación para los datos de entrada.

En esta etapa se debe tener en cuenta lo mencionado en 8.5.2. Aplicamos distintos criterio de filtrado por muestra en busca de los valores mas representativos. Lo anterior es importante ya que las señales de EEG son de naturaleza no estacionaria lo que quiere decir que sus componentes en frecuencia son variables con el tiempo y en ocasiones se puede considerar muestras que nos son realmente representativas. Además, el uso de esta técnica permite tener un marco de comparación con respecto a los distintos modelos observados. Los valores filtrados no serán eliminados de la muestra analizada, sino que se concentrarán en un nuevo grupo llamado Sin Clase que abarcará muestras no representativas, y tiempos muertos.

En este marco de comparación se realizan 4 pruebas por modelo.

1. Valores por debajo del desvío estándar.
2. Valores por debajo del Q3.
3. Valores por debajo del Q1.
4. Valores por debajo del valor medio.
5. Valores por debajo de la mediana.

Lista de referencia para resultados posibles:

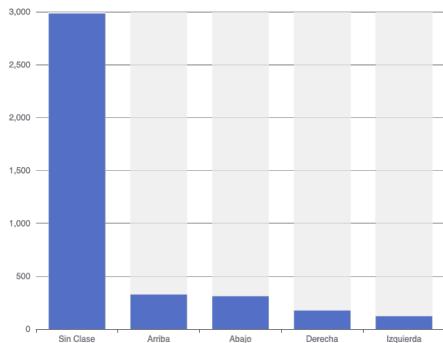
1. Sin Clase: es el resultado obtenido a partir de una muestra de entrada no representativa. Un tiempo muerto o ruido de la señal provista por el casco.
2. Arriba: es el resultado obtenido a partir de una muestra de entrada que genera un movimiento hacia arriba.
3. Abajo: es el resultado obtenido a partir de una muestra de entrada que genera un movimiento hacia abajo.
4. Izquierda: es el resultado obtenido a partir de una muestra de entrada que genera un movimiento hacia izquierda.
5. Derecha: es el resultado obtenido a partir de una muestra de entrada que genera un movimiento hacia derecha.

Criterio de lectura matrices de confusión: Las intersecciones de la diagonal principal contiene la cantidad de predicciones correctas por clase. Los valores ubicados en la fila representan la clase esperada, y los valores de las columnas representan las predicciones obtenidas. Los valores encontrados fuera de la diagonal principal representan falsos positivos.

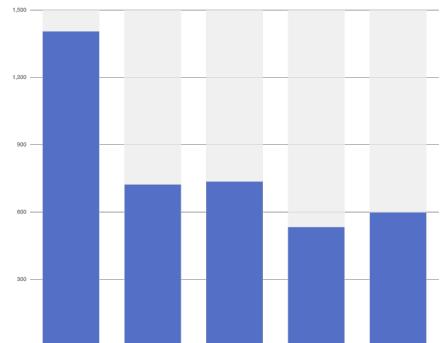
A continuación daremos detalle de las pruebas realizadas.

8.7.1. Regresión lineal simple

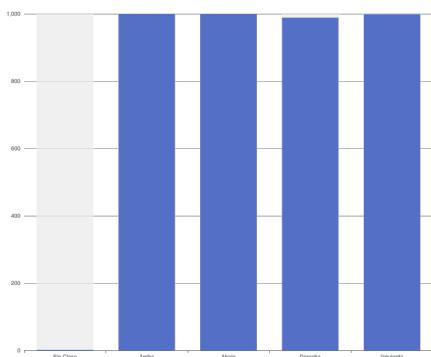
Los siguientes gráficos son representaciones de como quedan distribuidas las muestras una vez incorporadas al set de datos la clase de valores no representativos.



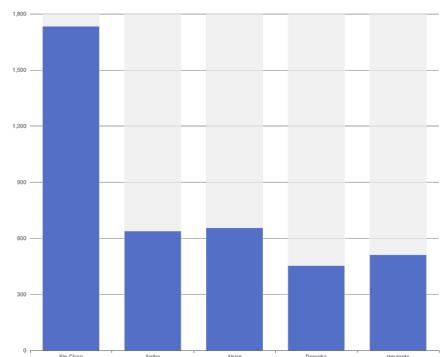
(a) Filtro y re-mapeo en base al desvío estándar.



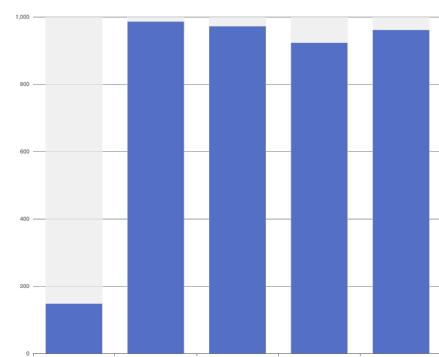
(b) Filtro y re-mapeo en base al Q3.



(c) Filtro y re-mapeo en base al Q1.



(d) Filtro y re-mapeo en base a la media.



(e) Filtro y re-mapeo en base a la mediana.

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	0	57	1151	0	0
Arriba	153	167	351	13	529
Abajo	169	238	274	37	429
Izquierda	153	197	365	10	498
Derecha	136	281	391	10	391

Cuadro 8.1: Matriz de confusión a partir de 8.6(a) precisión obtenida 14.03 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	1200	0	0	0	0
Arriba	607	154	137	0	266
Abajo	617	265	119	0	219
Izquierda	845	177	67	0	163
Derecha	730	143	127	0	164

Cuadro 8.2: Matriz de confusión a partir de 8.6(b) precisión obtenida 27.28 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	0	0	0	0	1177
Arriba	109	133	266	457	281
Abajo	124	73	236	411	342
Izquierda	100	111	145	335	495
Derecha	109	207	133	355	401

Cuadro 8.3: Matriz de confusión a partir de 8.6(c) precisión obtenida 18.41 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	0	534	684	0	0
Arriba	0	279	468	124	310
Abajo	0	295	466	96	333
Izquierda	2	337	540	76	299
Derecha	0	539	336	50	232

Cuadro 8.4: Matriz de confusión a partir de 8.6(d) precisión obtenida 17.55 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	0	0	0	0	1190
Arriba	10	126	78	57	916
Abajo	7	130	82	54	903
Izquierda	8	70	76	30	1026
Derecha	5	89	66	42	1035

Cuadro 8.5: Matriz de confusión a partir de 8.6(e) precisión obtenida 21.22 %

Del conjunto de matrices de confusión obtenido, se aprecia claramente que para el modelo de regresión lineal simple existe una gran aleatoriedad en los resultados. También se aprecia que existe un alto grado de clasificar una clase, cuando en realidad se esperaba otra.

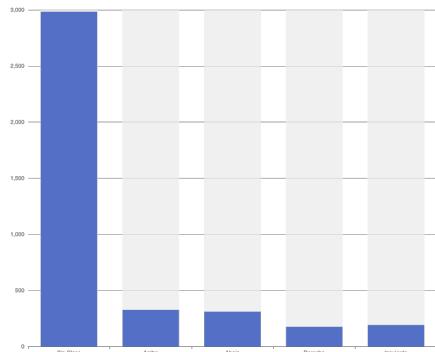
En muchos casos, se aprecia una clara tendencia por alguna clase en particular, descartando la existencia de otra, lo que claramente es erróneo para el problema analizado.

La precisión varió entre 14.03 % y 27.28 %, siendo esta en todos los casos demasiado baja para poder ser considerado como un modelo eficiente.

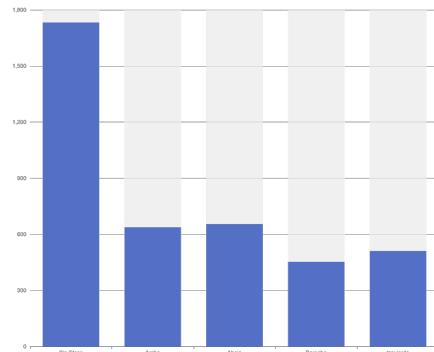
También se observa que 8.6(b) pierde por completo resultados para la clase izquierda. En tanto 8.6(d) a pesar de no ser el modelo de mayor precisión general es el que mas resultados esperados genera.

8.7.2. Naive Bayes

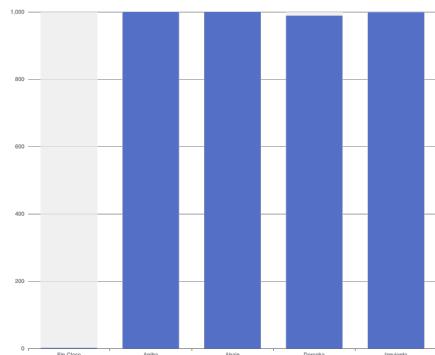
Los siguientes gráficos son representaciones de como quedan distribuidos los datos una vez incorporada la clase de valores no representativos.



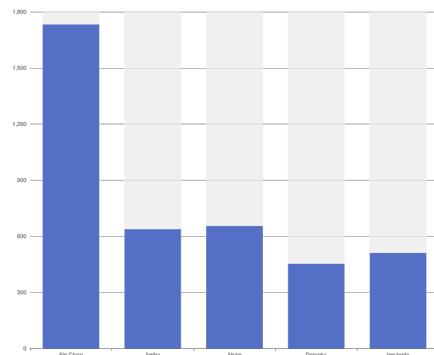
(f) Filtro y re-mapeo en base al desvío estándar.



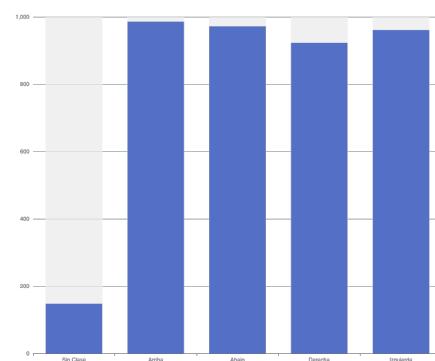
(g) Filtro y re-mapeo en base al Q3.



(h) Filtro y re-mapeo en base al Q1.



(i) Filtro y re-mapeo en base a la media.



(j) Filtro y re-mapeo en base a la mediana.

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	1019	56	4	19	49
Arriba	36	364	160	302	402
Abajo	42	250	255	274	397
Izquierda	23	259	156	432	333
Derecha	66	201	145	267	489

Cuadro 8.6: Matriz de confusión a partir de 8.6(f) precisión obtenida 42.65 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	981	14	5	61	148
Arriba	196	263	130	282	332
Abajo	140	188	175	316	367
Izquierda	186	172	114	318	399
Derecha	225	159	99	284	446

Cuadro 8.7: Matriz de confusión a partir de 8.6(g) precisión obtenida 36.38 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	12050	0	0	0	0
Arriba	0	178	207	365	469
Abajo	0	181	246	342	398
Izquierda	0	120	153	493	457
Derecha	0	113	138	387	548

Cuadro 8.8: Matriz de confusión a partir de 8.6(h) precisión obtenida 44.5 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	997	26	23	93	89
Arriba	214	242	127	352	267
Abajo	186	186	176	329	298
Izquierda	192	177	99	428	294
Derecha	265	151	107	332	350

Cuadro 8.9: Matriz de confusión a partir de 8.6(i) precisión obtenida 36.55 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	1063	3	4	31	72
Arriba	86	194	206	380	404
Abajo	87	174	246	361	337
Izquierda	113	130	154	414	352
Derecha	171	104	153	362	399

Cuadro 8.10: Matriz de confusión a partir de 8.6(j) precisión obtenida 38.6 %

Del conjunto de matrices de confusión observados, se aprecia una mejora en la precisión para el modelo de naive bayes respecto a la regresión lineal simple. También se aprecia que existe un alto grado de clasificar una clase, cuando en realidad se esperaba otra.

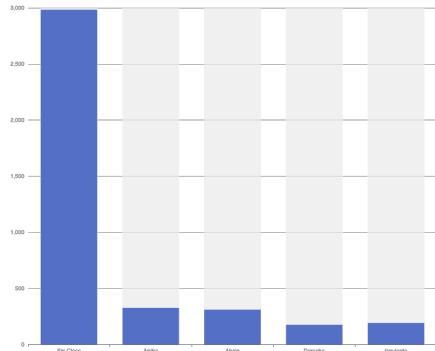
No descarta clases, lo que muestra un indicio de que el modelo se ajusta mejor al problema planteado.

La precisión varió entre 36.38 % y 44.5 %, siendo esta en todos los casos demasiado baja para poder ser considerado como un modelo eficiente.

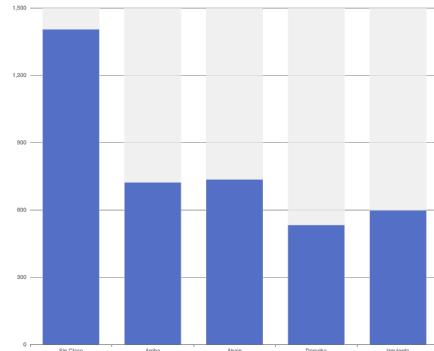
También se observa que 8.8 siendo el modelo de mayor precisión general es el que mas resultados esperados genera a pesar de que muchos de los resultados estén clasificados dentro de la clase de resultados no representativos, es el que menor cantidad de valores erróneos predijo.

8.7.3. Árbol de decisión

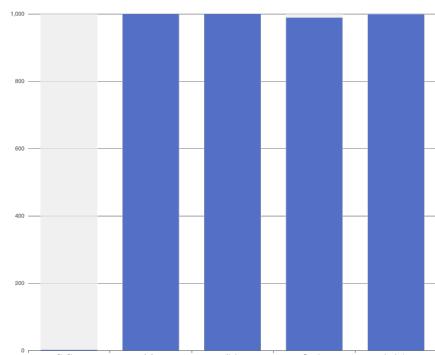
Los siguientes gráficos son representaciones de como quedan distribuida la muestra una vez incorporada al set de datos la clase de valores no representativos.



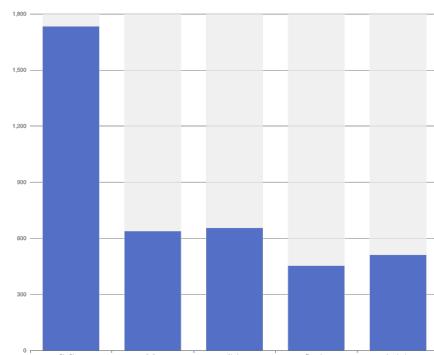
(k) Filtro y re-mapeo en base al desvío estándar.



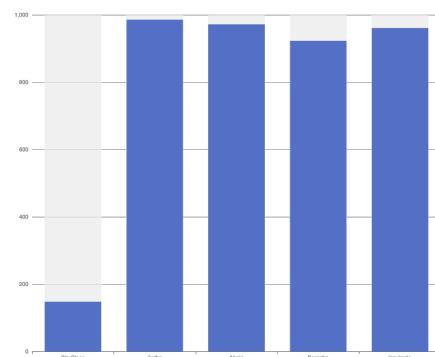
(l) Filtro y re-mapeo en base al Q3.



(m) Filtro y re-mapeo en base al Q1.



(n) Filtro y re-mapeo en base a la media.



(ñ) Filtro y re-mapeo en base a la mediana.

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	976	36	74	61	55
Arriba	40	327	253	295	235
Abajo	32	256	397	372	196
Izquierda	46	261	245	437	238
Derecha	42	247	242	310	327

Cuadro 8.11: Matriz de confusión a partir de 8.6(k) precisión obtenida 41.07 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	793	64	77	146	175
Arriba	76	398	222	204	244
Abajo	62	324	407	180	201
Izquierda	131	294	238	341	216
Derecha	141	278	231	199	358

Cuadro 8.12: Matriz de confusión a partir de 8.6(l) precisión obtenida 38.28 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	1170	0	0	0	0
Arriba	0	325	430	230	243
Abajo	0	192	589	207	206
Izquierda	0	186	305	447	269
Derecha	0	217	325	298	369

Cuadro 8.13: Matriz de confusión a partir de 8.6(m) precisión obtenida 48.33 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	796	41	71	119	188
Arriba	205	318	254	196	221
Abajo	207	244	347	206	213
Izquierda	221	188	185	319	220
Derecha	236	199	221	225	360

Cuadro 8.14: Matriz de confusión a partir de 8.6(n) precisión obtenida 35.67 %

Clase	Sin Clase	Arriba	Abajo	Izquierda	Derecha
Sin Clase	1095	21	16	51	49
Arriba	49	473	225	252	221
Abajo	30	353	424	248	171
Izquierda	86	270	183	412	199
Derecha	80	290	204	302	296

Cuadro 8.15: Matriz de confusión a partir de 8.6(ñ) precisión obtenida 45 %

Del conjunto de matrices de confusión observado, se aprecia una mejora en la precisión con respecto a los anteriores 2 modelos analizados siendo la misma hasta 4 % mayor que la mejor medida obtenida hasta el momento.

Al igual que para 8.7.2, el 8.13 obtuvo el mejor desempeño. La diagonal principal de la matriz que indica los valores correctamente predichos tiene la mayoría de los resultados, y la cantidad de valores predichos erráticamente es menor al promedio del resto de las matrices.

Al igual que Bayes, no descarta clases, lo que muestra un indicio de que el modelo se ajusta mejor al problema planteado.

La precisión varió entre 35.67 % y 48.33 %, casualmente, a pesar de tener métodos de filtrado diferente, los modelos de árbol de decisión en general son los mas precisos ante el problema planteado obteniendo en promedio 41.67 % de asersión, 1.93 % mejor que el promedio de 39.73 % de Bayes, y casi 22 % mejor que el promedio de 19.69 % arrojado por la regresión lineal simple. Aún así, la precisión promedio acusa un mal rendimiento del modelo ante el problema planteado.

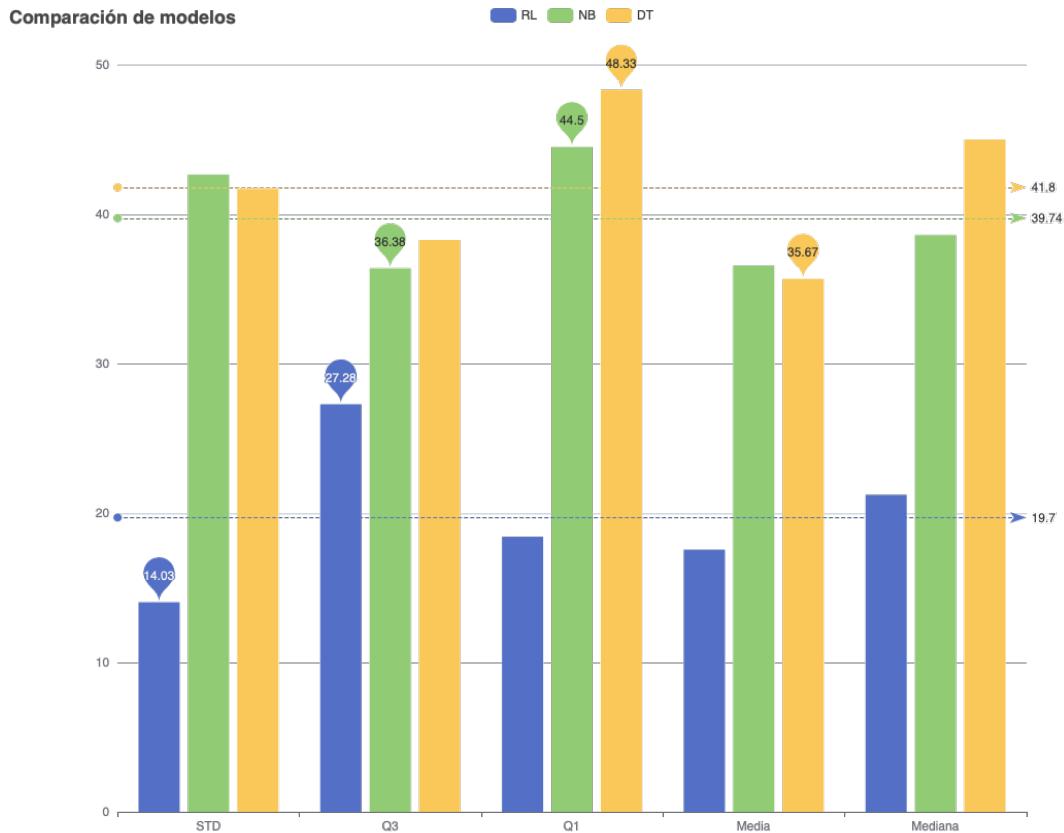


Figura 8.6: Comparación de precisión obtenida al generar los modelos evaluados en base a los distintos criterios de filtros utilizados.

En base a los resultados obtenidos 8.6, considerando el problema y el trabajo realizado sobre el procesamiento de los datos, se procede a probar la creación del modelo pero con mayor volumen de datos, esperando que mientras mayor nivel de conocimiento tuviera el algoritmo mejor podría llegar a predecir. Para estas pruebas, se decidió tomar como base el árbol de decisión, ya que presentó mejor rendimiento promedio que los demás modelos observados. Los resultados obtenidos se muestran

a continuación:

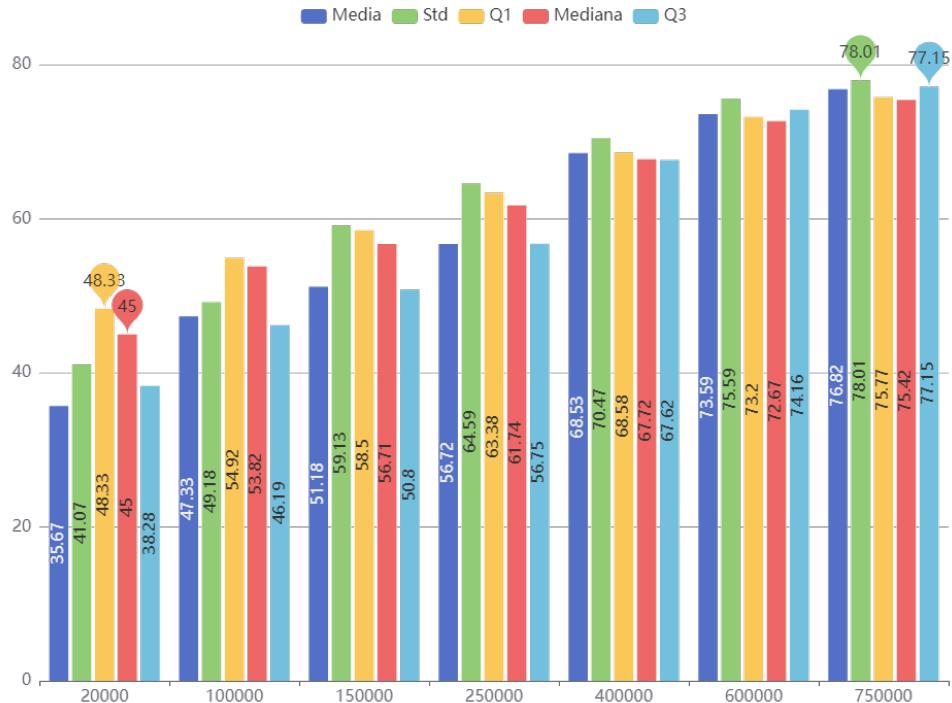


Figura 8.7: Evolución de precisión de árbol de decisión en función de la cantidad de muestras.

Observamos que en principio, con una cantidad total de 20000 muestras (4000 por clase), el remapeo por Q1, como se mencionó anteriormente, es el que produce mayor precisión, seguido por la mediana. Pero conforme aumentamos la cantidad de muestras, el remapeo por desvío estándar gana terreno frente a otras medidas, seguido por el remapeo por Q3. Llegando a un total de 750000 muestras (150000 por clase), el desvío estándar obtiene el primer lugar con una precisión de 78.01 %, mientras que Q3 alcanza una precisión de 77.15

Con todo el análisis presentado hasta el momento, queda claro que el modelo seleccionado para esta tesina es el modelo de árbol de decisión.

8.8. Pruebas de modelo en tiempo real

Con el modelo entrenado, y habiendo obtenido valores de precisión lo suficientemente altos como para tener confianza sobre su desempeño, surge un nuevo desafío, evaluar una nueva muestra.

Para entender el flujo de trabajo al momento de analizar valores en tiempo real se debe entender como se procesan los nuevos valores. Luego se comentará como se incorpora el casco, y las pruebas realizadas.

8.8.1. Procesamiento de nuevos valores

El primer paso fue incorporar al motor de toma de datos un nuevo servicio que fuera capaz de tomar el último valor censado por el casco, procesarlo, y pasárselo la información al modelo ya entrenado.

Se separa la clasificación en base al modelo del pre procesamiento, en donde se procede a realizar los siguientes pasos:

1. Normalizar : en base al modelo entrenado se obtienen los valores mínimos y máximos considerados.
2. FFT: se aplica de manera horizontal sobre los valores de la muestra.
3. Matriz de desvíos: se calcula utilizando los valores del set de datos entrenados en el conjunto inicial.
4. Truncado: se redondean los valores numéricos a 5 decimales para obtener una mejor precisión.

De esta forma, los datos evaluados en tiempo de ejecución tienen las mismas características que los datos con los que se evaluó el modelo inicial, por tanto son comparables y clasificables en base a dicho conjunto.

A continuación la muestra preprocesada ingresa al modelo de clasificación que dará un movimiento entre los evaluados en base a los valores de la muestra.

8.8.2. Uso del casco

Para probar el modelo en tiempo real, fue indispensable contar con una herramienta que bajo concentración, nos permita representar los diferentes movimientos clasificados por el modelo. En este punto se introduce el laberinto 7.3.3.

Para realizar las pruebas, se considera mantener el ambiente en silencio, sin dispositivos móviles que puedan afectar nuestra atención y alterar las ondas cerebrales tomadas por el dispositivo, enfocados únicamente en generar los movimientos para resolver el “juego”.

El método utilizado es muy sencillo de explicar, pero no tan fácil de aplicar. En su turno, cada participante realizó los siguientes pasos:

1. Se situó frente a la computadora que actúa de servidor.
2. Colocó el casco sin prenderlo aún.
3. Conectó el Dongle USB al servidor.
4. Inicializó todos los servicios asociados(ThinkGear Connector, Conecador Node.js para lectura de datos reales, Laberinto).
5. Prendió el casco.
6. Abrió la URL del laberinto, habilitando la toma de datos en tiempo real.

8.8.2.1. Observaciones

Se realizaron múltiples intentos llegando a sortear todos los obstáculos en más de una ocasión.

El ritmo de predicción en entorno real resulta variable. La completitud del laberinto presentado en 7.3.3 no estuvo exenta de errores de predicción por lo que se observan idas y vueltas durante la ejecución. Esto produjo tiempos de resolución del juego de alrededor de 10 minutos. Como se mencionó anteriormente, un ambiente calmo que permita la concentración del usuario es de suma importancia. Los ambientes ruidosos afectan la forma de las lecturas tomadas, lo cual confunde al algoritmo de predicción.

Otro tema a considerar son los problemas técnicos que presenta el modelo de casco utilizado. Este casco es una versión antigua, por lo cuál funciona mediante una pila de tipo AAA, también requiere que las mismas estén completamente cargadas continuamente. Esto además de ser un gasto insostenible en el tiempo, provoca problemas de conexión cuando las pilas se descargan por el uso continuado, tanto durante las pruebas con el laberinto como durante la etapa de desarrollo de las herramientas y toma de sets de pruebas.

Existen otros problemas de conexión que ocurren ocasionalmente: aún siguiendo los manuales de desarrollo y uso provistos por el fabricante y contando con pilas cargadas, el casco no puede conectarse por motivos desconocidos. De todas maneras remarcamos que estos son problemas de este dispositivo particular y su fabricación y funcionamiento.

8.9. Resumen

Este capítulo detalla la experiencia concreta de trabajo a partir de las herramientas mencionadas en el capítulo 7, desde el momento de la obtención datos hasta la experiencia de prueba de tiempo real y uso del casco. Se detallan las etapas de obtención de datos y pre-procesamiento, aquellas que representaron el mayor tiempo de desarrollo del proyecto y como tales las que determinaron la concreción de resultados. A continuación se detalla los modelos de machine learning utilizados para el proyecto con el análisis detallado y comparativo de los de resultados que obtuvimos en etapa de pruebas. Para finalizar se repasa brevemente la experiencia al usar el casco EEG en vivo para la resolución del ejercicio de prueba del laberinto.

En el siguiente capítulo se exponen conclusiones finales y aquellos proyectos que podrían suceder a este en un futuro.

Capítulo 9

Conclusiones y trabajos futuros

9.1. Conclusiones

Habiendo completado este proyecto de tesina, se puede llegar a diversas conclusiones sobre los objetivos que se han planteado. El objetivo principal se considera cumplido. Se ha logrado implementar un prototipo que, utilizando un casco de lectura EEG comercial, no más invasivo que un set de auriculares, logra obtener lecturas de actividad cerebral y convertirlas en un formato de uso ampliamente conocido, lo que permite llevar la información a otras herramientas para visualización y proceso.

En relación al uso del casco de lectura de ondas cerebrales y tecnologías BCI, si bien durante las pruebas, este modelo en particular presentó inconvenientes técnicos, la tecnología tiene mucho potencial de uso para desarrollar herramientas que ayuden a mejorar la calidad de vida humana. Se recomienda continuar probando desarrollos similares al presentado en este proyecto, empleando hardware alternativo más actualizado y con métodos de conexión más prácticos a nivel de implementación.

Durante el transcurso del proyecto, y habiendo elegido JavaScript como lenguaje de programación principal, se pudo comprobar de primera mano su enorme versatilidad. Permitió desarrollar soluciones para diversos ámbitos de manera ágil. No obstante, se observó que a pesar de brindar una base suficiente para crear modelos de Machine Learning aptos, el ecosistema aún está lejos de igualar el nivel de desarrollo y soporte que brindan otros lenguajes de programación más enfocados en este ámbito. Para la solución implementada fue necesario desarrollar herramientas que en otros lenguajes se encuentran disponibles de base. También usar librerías aisladas con distinto nivel de desarrollo que en otros lenguajes se encuentran unificadas.

Con respecto a la utilidad de las herramientas, se puede concluir que en las condiciones actuales estas no resultan utilizables en entornos del mundo real de manera inmediata. Sin embargo, el análisis y las pruebas realizadas demuestran que son una base firme para desarrollos futuros.

9.2. Trabajos futuros

A fin de continuar indagando en las posibilidades que este proyecto sugiere, se puede trabajar en las siguientes ideas:

- Desarrollar un método de muestreo que no tome muestras individuales de lectura, sino, muestras sucesivas extendidas en el tiempo, puesto que el proceso cognitivo del movimiento puede verse mejor representado por múltiples muestras en una fracción de tiempo que muestren una tendencia, en vez de una muestra en un momento dado.
- Mejorar el pre-procesamiento de los set de datos de entrenamiento, exponiendo con mayor intensidad la correlación entre las mediciones dentro una misma muestra mediante otro tipo de medidas estadísticas.
- Adaptar la solución desarrollada en este proyecto a un modelo más avanzado del mismo dispositivo, otros modelos del mismo fabricante o incluso de otros fabricantes, siempre y cuando el mecanismo de lectura no difiera significativamente.
- Desarrollar una funcionalidad que permita mejorar el entrenamiento del modelo de machine learning utilizado de manera progresiva, tomando nuevas muestras durante la utilización del ejercicio del laberinto propuesto como mecanismo de pruebas, a fin de aumentar la cantidad de observaciones disponibles.
- En caso de lograr mejores resultados con alguna de las propuestas mencionadas (U otras alternativas), aplicar el funcionamiento desarrollado para este proyecto a experimentos más avanzados, por ejemplo, utilizando Arduino para pruebas que conduzcan a nuevos resultados.
- Lograr una versión de la solución propuesta publicable en la nube, que permita a un usuario conectarse a un sitio web, y enviar las lecturas del casco EEG a ese servidor.

Apéndice A

Guías de uso

A.1. Introducción

A continuación se detallan guías de uso básicas para las aplicaciones desarrolladas para los procesos de investigación este proyecto de tesina. El detalle de la implementación de las mismas puede verse en la sección 7.3.

A.2. Prerequisitos

A.2.1. General

Las aplicaciones que componen la solución se desarrollaron sobre Sistema Operativo Windows 10. Para su utilización requieren la instalación del software de conexión del casco: ThinkGear Connector. En segundo lugar, se requiere la instalación del runtime Node.js.

A.2.2. Software Thinkgear Connector

Este software se puede obtener desde las siguientes URL:

- Sitio:http://developer.neurosky.com/docs/doku.php?id=thinkgear_connector_tgc
- Software:<http://neurosky.fetchapp.com/permalink/a382ab>

El software requiere la instalación del framework .NET 4.0, que puede obtenerse en la siguiente URL:

<http://www.microsoft.com/en-us/download/details.aspx?id=17851>

A.2.2.1. Instalación

Una vez descargados los instaladores, instalar el framework 4.0 en primer lugar. Se trata de un instalador sencillo, salvo que haya alguna particularidad del equipo destino, con seguir las instrucciones en pantalla, el proceso se completa en unos minutos.

La instalación de ThinkGear Connector esta incluida en un comprimido. Es necesario descomprimir el archivo, y dentro del mismo encontraremos la carpeta ThinkGear Connector, con el siguiente contenido.

Usuarios > luism > Descargas > ThinkGear_Connector > ThinkGear_Connector		
Nombre	Tipo	Tamaño
zh-CHS	Carpeta de archivos	
zh-CHT	Carpeta de archivos	
BrainDb.dll	Extensión de la aplicación	12 KB
Connector.dll	Extensión de la aplicación	25 KB
Jayrock.Json.dll	Extensión de la aplicación	132 KB
NLog.config	XML Configuration File	1 KB
NLog.dll	Extensión de la aplicación	297 KB
PacketFormatter.dll	Extensión de la aplicación	13 KB
ThinkGear Connector.exe	Aplicación	446 KB
ThinkGear Connector.exe.config	XML Configuration File	1 KB
ThinkGear.dll	Extensión de la aplicación	179 KB
WinSparkle.dll	Extensión de la aplicación	922 KB

Figura A.1: Contenido de instalación de ThinkGear Connector

Esto es una instalación portable, por lo cual se puede ejecutar desde cualquier ubicación. Aún así, es recomendable ubicarla en un directorio apropiado, por ejemplo: C:\Program Files (x86)\NeuroSky\ThinkGear Connector. Hecho esto, crear un acceso directo para el archivo “ThinkGear Connector.exe”, para mayor practicidad.

A.2.3. Node.js

Node.js es el runtime sobre el cual se ejecutan todas las aplicaciones de la solución.

A.2.3.1. Instalación de Node.js

La instalación básica será suficiente, por lo tanto, se deberá descargar la versión estable más reciente, e instalar siguiendo las instrucciones provistas por el instalador. Su instalador se puede descargar desde el siguiente link:

<https://nodejs.org/en/>

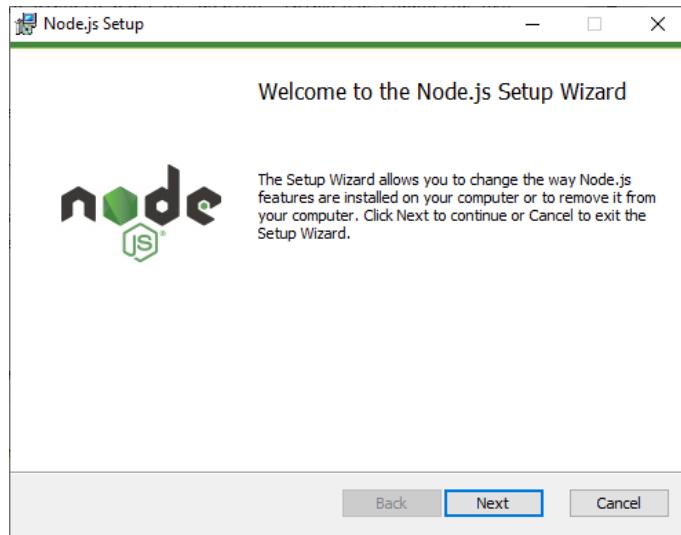


Figura A.2: Instalador de Node.js

A.2.3.2. Instalación de aplicaciones

Para poder ejecutar las aplicaciones Node.js, es necesario ejecutar el comando *install* de Node.js para cada una de ellas. Este comando se encarga de descargar desde la nube las dependencias necesarias para ejecutar cualquier aplicación implementada en Node.js, en base a lo definido en el archivo *package.json* incluido en cada aplicación. Este paso requiere una conexión a internet, puesto que las dependencias se obtienen de repositorios en la nube.

Las dependencias no se incluyen en el repositorio de las aplicaciones por conveniencia de la comunidad de desarrollo de Node.js, debido a que ocupan gran cantidad de espacio en disco. Así alivian los tipos de carga y descarga del código de cada aplicación.

A.2.3.3. Pasos

- Abrir una terminal o consola de comandos o powershell de windows, y moverse al directorio del código fuente de la aplicación.
- Usar el comando: *npm install*

A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command "PS C:\> cd C:\ruta\origen\node_app_dir" is entered, followed by "PS C:\ruta\origen\node_app_dir> npm install". The output shows the command being processed.

Figura A.3: Comando de instalación de aplicación Node.js

- Como resultado de este proceso, en el directorio de cada aplicación se creará una nueva carpeta llamada *node-modules*, que contendrá binarios y archivos de código de extensión *js*.

A.3. Conector Node.js para lectura de datos reales

El código fuente asociado a esta aplicación se encuentra disponible en el siguiente repositorio GIT en Github.com:

<https://github.com/luisluna-arg/TesinaLicInformaticaUNPJSB>

Subdirectorio “CoreEngine”

A.3.1. Requerimientos

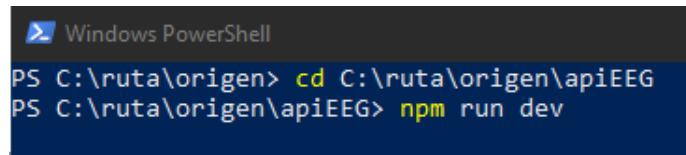
- ThinkGearConnector (Ver A.2.2).

A.3.2. Uso de API

1. Previo a la utilización de la aplicación es necesario configurar las credenciales de acceso a Firestore. Para ello es necesario los siguiente:
 - a) Ubicar dentro del directorio del código fuente de la aplicación el siguiente archivo de configuración: “src\config\firebase.config.js”
 - b) Agregar en el archivo los siguientes datos de conexión, completando cada campo con su correspondiente dato:
 - apiKey: “AIzaSyDQfJLhblyYblxDU6AWrDgDETWWxRSPMDY”
 - authDomain: “tesina-329921.firebaseio.com”
 - databaseURL: “https://tesina-329921-default-rtdb.firebaseio.com”
 - projectId: “tesina-329921”
 - storageBucket: “tesina-329921.appspot.com”
 - messagingSenderId: “216467805959”
 - appId: “1:216467805959:web:dc35a7a495f70522d0d7a2”
 - measurementId: “G-F7DGTNBQ5Z”

Nota: Estos campos se encontraran vacíos en el código del repositorio en Github. No es una buena práctica almacenar credenciales en Github.

2. Ejecutar la aplicación ThinkGearConnector de NeuroSky
3. Iniciar el servidor de la siguiente manera:
 - Abrir una terminal, consola o powershell de windows, y moverse al directorio del código fuente del Conector de lectura de datos.
 - Usar el comando: *npm run dev*
4. Conectar el Dongle USB del casco EEG a la computadora
5. Colocarse el casco EEG y encenderlo. Espere hasta que tanto en el Dongle USB como el casco EEG se encienda una luz, que pasa de rojo a azul, esto indicará que el dispositivo se ha conectado correctamente.
6. Esperar un momento y la API EEG debería comenzar a tomar lecturas de ondas cerebrales.



```

Windows PowerShell

PS C:\ruta\origen> cd C:\ruta\origen\apiEEG
PS C:\ruta\origen\apiEEG> npm run dev

```

Figura A.4: Comando de ejecución de Conector para consulta de datos de lectura



Figura A.5: Dongle USB de conexión de casco “MindWave MW001”



Figura A.6: Dongle USB y casco EEG conectados de forma correcta

A.3.3. Métodos de la API

Una vez que el casco EEG está conectado, la API puede utilizarse mediante peticiones HTTP utilizando alguno de los siguientes métodos para acceder a las lecturas de ondas cerebrales:

- all: Devuelve todas las muestras leídas durante la sesión actual o una cantidad específica mediante el parámetro size.
Ej.: localhost:xxxx/all?size=5
- single: Devuelve la lectura más reciente del casco EEG.
Ej.: localhost:xxxx/single
- getDirection: Devuelve una predicción de dirección basada en la lectura EEG más reciente.
Ej.: localhost:xxxx/getDirection
Las direcciones son números enteros que van desde 0 a 4:
 - 0 Sin movimiento
 - 1 Abajo
 - 2 Arriba
 - 3 Izquierda
 - 4 Derecha

A.3.4. Consideraciones en caso de problemas de conexión del casco

Estos son algunas situaciones a tomar en cuenta en caso de no lograr una buena conexión con el casco:

- Pilas descargadas
- Contacto pobre de los sensores o electrodos con la piel de la persona (El pelo interrumpe el contacto, el casco no se ajusta a la cabeza del usuario o no esta colocado correctamente).
- El usuario se mueve excesivamente.
- Ruido por electro estática excesiva en el ambiente (Señales eléctricas fuertes o mucha carga estática en el usuario).
- Excesivo ruido biométrico no EEG (Por ejemplo: EMG, EKG/ECG, EOG, etc.)

Tomado de [18]

A.4. Gráfico en vivo

El código fuente asociado a esta aplicación se encuentra disponible en el siguiente repositorio GIT en Github.com:

<https://github.com/luisluna-arg/TesinaLicInformaticaUNPJSB>

Subdirectorio “EEGLiveChart”

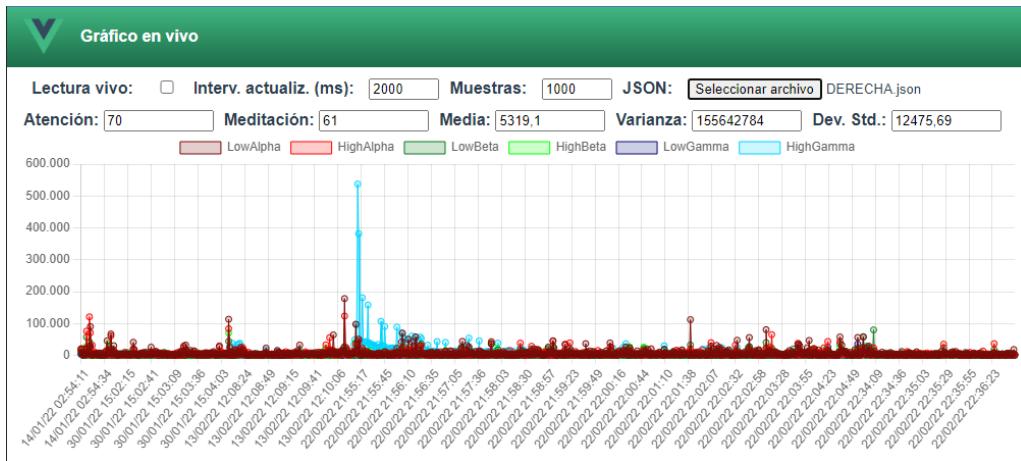


Figura A.7: Aplicación de visualización de gráficos en vivo.

A.4.1. Inicio de la aplicación

Esta aplicación puede funcionar de forma complementaria con el Conector para lectura de datos mencionado en A.3, mostrando datos capturados en el momento, y también mostrando información en crudo a partir de archivos de tipo JSON.

A.4.1.1. Pasos

1. Iniciar el Conector Node.js para lectura (Ver A.3).
2. Iniciar la aplicación web de la siguiente manera:
 - Abrir una terminal, consola o powershell de windows, y moverse al directorio del código fuente de la web Laberinto.
 - Usar el comando: `npm run serve`

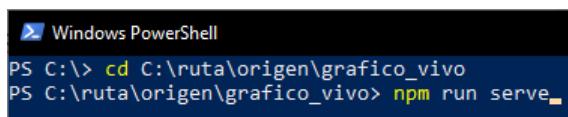


Figura A.8: Comando de ejecución de aplicación web de Gráfico en vivo

- Iniciada la aplicación, la terminal mostrará la URL de la web, de forma similar a la siguiente:

```

Seleccionar C:\Windows\system32\cmd.exe
DONE Compiled successfully in 12246ms

App running at:
- Local: http://localhost:8081/
- Network: http://192.168.0.15:8081/

Note that the development build is not optimized.
To create a production build, run npm run build.

```

Figura A.9: Aplicación web de Gráfico en vivo iniciada correctamente

A.4.2. Uso de la aplicación

La aplicación web permite las siguientes funcionalidades:

- **Controles de lectura en vivo:** Estos controles permiten activar o desactivar la lectura de actividad cerebral mediante el casco EEG. Se cuenta con tres controles:
 1. *Checkbox lectura en vivo:* Activa o desactiva la lectura en vivo.
 2. *Intervalo de actualización:* Determina el tiempo transcurrido en milisegundos entre cada petición de lecturas.
 3. *Muestras:* Cantidad de muestras a recibir en cada petición por intervalos.
- **Carga de archivo JSON:** Permite cargar un archivo JSON con datos en crudo obtenidos mediante el Conector de lectura de datos, y verlo en pantalla de la misma manera en que se vería una lectura en vivo.

Lectura vivo:	<input type="checkbox"/>	Interv. actualiz. (ms):	2000	Muestras:	1000	JSON:	<input type="button" value="Seleccionar archivo"/>	Ninguno archivo selec.
Atención:	70	Meditación:	61	Media:	5319,1	Varianza:	155642784	Dev. Std.: 12475,69

Figura A.10: Barra de herramientas de aplicación de gráficos en vivo.

- **Valores estadísticos:** La aplicación cuenta con campos de datos estadísticos. Estos valores se calcularán de forma inmediata, tanto al mostrar lecturas en vivo como al mostrar información mediante archivos de tipo JSON. Los valores presentados son: atención, meditación, media, varianza y desvío estándar.
- **Selección de líneas de tendencia:** Sobre el gráfico en vivo se muestran las leyendas de cada línea de tendencia. Al presionar el nombre de cada una, es posible mostrarla u ocultarla.

A.5. Laberinto

El código fuente asociado a esta aplicación se encuentra disponible en el siguiente repositorio GIT en Github.com:

<https://github.com/luisluna-arg/TesinaLicInformaticaUNPJSB>

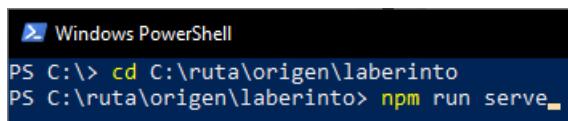
Subdirectorio “EEGMaze”

A.5.1. Inicio de la aplicación

Esta aplicación funciona de forma complementaria con el Conector para lectura de datos. Utilizando el método de predicción explicado anteriormente, la aplicación permite mover un puntero a fin de resolver un laberinto, utilizando sólo la actividad cerebral del jugador.

A.5.1.1. Pasos

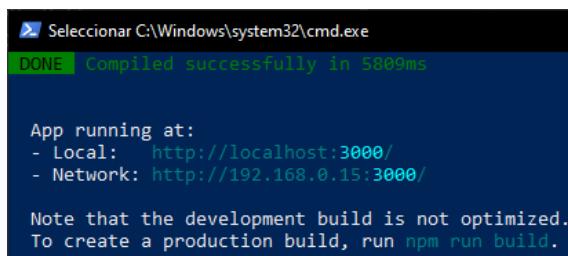
1. Iniciar el Conector Node.js para lectura (Ver A.3).
2. Iniciar la aplicación web de la siguiente manera:
 - Abrir una terminal, consola o powershell de windows, y moverse al directorio del código fuente de la web Laberinto.
 - Usar el comando: `npm run serve`



```
Windows PowerShell
PS C:\> cd C:\ruta\origen\laberinto
PS C:\ruta\origen\laberinto> npm run serve
```

Figura A.11: Comando de ejecución de aplicación web de Laberinto

- Una vez iniciada la aplicación, la terminal mostrará la URL de la web, de forma similar a la siguiente:



```
Seleccionar C:\Windows\system32\cmd.exe
DONE Compiled successfully in 5809ms

App running at:
- Local: http://localhost:3000/
- Network: http://192.168.0.15:3000/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

Figura A.12: Aplicación web de Laberinto iniciada correctamente

A.5.2. Uso de la aplicación

La aplicación web cuenta con dos funcionalidades para interacción del usuario:

- **Activación de lectura en vivo:** Este checkbox permite activar o desactivar la lectura de actividad cerebral mediante el casco EEG.

Lectura en vivo:

Figura A.13: Checkbox de lectura en vivo de Laberinto

- **URL de Servidor:** Este campo de texto permite modificar en vivo la dirección URL en la cual se ubica la Conector Node.js para lectura de datos. Útil en el caso de que el Conector se aloje en una dirección distinta de la dirección por defecto.

URL Servidor:

Figura A.14: Caja de texto para URL de servidor de Conector Node.js para lectura de datos

A.5.3. Instrucciones de juego

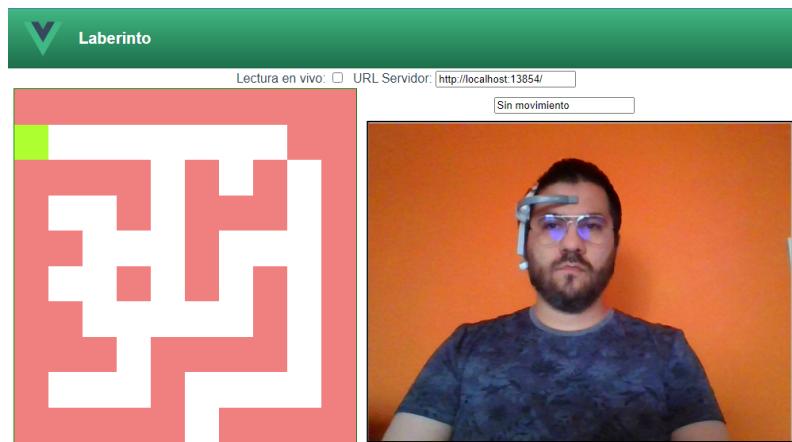


Figura A.15: aplicación web de laberinto en ejecución

Una vez activada la lectura en vivo, puede procederse a resolver el laberinto. Es importante para mejorar la efectividad del juego contar con un ambiente calmo, sin interrupciones, que permita al jugador concentrarse en la tarea.

La mecánica es simple. El jugador deberá concentrarse en mover la celda verde del laberinto. Para ello, puede imaginarse el movimiento de esa celda hacia la dirección deseada. Es importante que el usuario haga el ejercicio de visualizar el movimiento en su mente, como si estuviera ocurriendo realmente.

Se ha probado que también es de utilidad generar actividad cerebral mediante el cierre de los puños, así como mover la cabeza, en ambos casos según la dirección deseada. Con menor certeza, se ha considerado que también es útil pensar o pronunciar la palabra de la dirección deseada.

Durante la ejecución puede verse una cámara en vivo del jugador (Con el fin de poder ver y capturar sus reacciones), así como una caja de texto que muestra a cada momento la dirección que se ha detectado en base a la actividad generada.

Glosario

BCI Brain computer interface, interfaz cerebro computadora.. 1, 5

código fuente Código de la implementación de un programa de computadora escrito en un lenguaje de programación, y que aún no ha sido procesado por un compilador o herramienta similar para generar el producto final.. 11

Dongle USB Conector USB similar a un pendrive que permite conectar algún dispositivo a una computadora. 43, 72, 79

EEG Electroencefalografía, examen que sirve para medir la actividad eléctrica del cerebro.. 1

Entropía Según el diccionario de la RAE, medida del desorden de un sistema. Una masa de una sustancia con sus moléculas regularmente ordenadas, formando un cristal, tiene entropía mucho menor que la misma sustancia en forma de gas con sus moléculas libres y en pleno desorden.. 38

framework Marco de Trabajo. En desarrollo de software, conjunto de herramientas de software desarrolladas para proveer un marco de trabajo estandarizado y reutilizable para situaciones comunes.. 48

Gini Es una medida de la desigualdad ideada por el estadístico italiano Corrado Gini.. 38

IA Inteligencia Artificial es la combinación de algoritmos planteados con el propósito de crear máquinas que presenten las mismas capacidades que el ser humano.. 1

overfitting Efecto de sobreentrenar un algoritmo de aprendizaje con unos ciertos datos para los que se conoce el resultado deseado.. 38

Bibliografía

- [1] Hyun Jae Baek, Min Hye Chang, Jeong Heo, and Kwang Suk Park. Enhancing the usability of brain-computer interface systems. *Computational Intelligence and Neuroscience*, 2019:1–12, Junio 2019. Article ID 5427154.
- [2] Simanto Saha1, Khondaker A. Mamun, Khawza Ahmed, Raqibul Mostafa, Ganesh R. Naik, Sam Darvishi1, Ahsan H. Khandoker, and Mathias Baumert. Progress in brain computer interface: Challenges and opportunities. *Computational Intelligence and Neuroscience*, 15:1–20, Febrero 2021. Article ID 578875.
- [3] Yongwook Chae, Sungho Jo, and Jaeseung Jeong. Brain-actuated humanoid robot navigation control using asynchronous brain-computer interface. In *2011 5th International IEEE/EMBS Conference on Neural Engineering*, pages 519–524, 2011.
- [4] Luis Vergara, Yongrui Huang, Jianhao Yang, Pengkai Liao, and Jiahua Pan. Fusion of facial expressions and eeg for multimodal emotion recognition. *Computational Intelligence and Neuroscience*, 2017:1–8, Septiembre 2017. Article ID 2107451.
- [5] Red Hat Inc. ¿qué es el open source? <https://www.redhat.com/es/topics/open-source/what-is-open-source>.
- [6] Dave Gray. 7 Key Benefits of Open Source Software. <https://www.developer.com/project-management/open-source-software-benefits>, 31 de Agosto de 2017.
- [7] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, Stan Bileschi, Michael Terry, Charles Nicholson, Sandeep N. Gupta, Sarah Sirajuddin, D. Sculley, Rajat Monga, Greg Corrado, Fernanda B. Viegas, and Martin Wattenberg. Tensorflow.js: Machine learning for the web and beyond. Technical report, Conference on Machine Learning and Systems, Arrillaga Alumni Center, Stanford University, Febrero 2019.
- [8] WebAssembly. <https://webassembly.org>. W3C Community Group.
- [9] OpenGL ES for the Web. <https://www.khronos.org/webgl/>. The Khronos Group.
- [10] About node.js. "<https://nodejs.org/en/about>", 2021. OpenJS Foundation.
- [11] Vue.js. sitio de documentación de vue.js: Guía para vue.js v2.x. <https://es.vuejs.org/v2/guide/>.

- [12] Evan You. Announcing vue 3.0 “One Piece”. <https://blog.vuejs.org/posts/vue-3-one-piece.html>, Septiembre 2020. E. You, creador y desarrollador principal de Vue.js.
- [13] Peter Stone, Rodney Brooks, Erik Brynjolfsson, Ryan Calo, Oren Etzioni, Greg Hager, Julia Hirschberg, Shivaram Kalyanakrishnan, Ece Kamar, Sarit Kraus, Kevin Leyton-Brown, David Parkes, William Press, AnnaLee Saxenian, Julie Shah, Milind Tambe, and Astro Teller. Artificial intelligence and life in 2030. Technical report, Stanford University, Stanford, CA, Septiembre 2016.
- [14] S.J. Russell and Norvig P.]. *Artificial Intelligence*. PEARSON-PRENTICE HALL, 2019.
- [15] George C. Canavos. *Probabilidad y Estadística*. McGraw-Hill/Interamericana de México, S.A. de C.V., 1988.
- [16] Osvaldo Martin. *Bayesian Analysis with Python - Second Edition*, chapter 1. Packt Publishing, O'Reilly, December 2018.
- [17] Wikimedia Commons. Este árbol de decisión representa el juego 'piedra, papel o tijera' y las diferentes opciones que pueden elegir sus jugadores. https://commons.wikimedia.org/wiki/File:%C3%81rbol_de_decisi%C3%B3n_del_juego_%27Piedra,_papel_o_tijera%27.png.
- [18] Inc. NeuroSky. Thinkgear serial stream guide. http://developer.neurosky.com/docs/doku.php?id=thinkgear_communications_protocol, Enero 2017.
- [19] Francisco Ariza-Lopez, José Rodríguez-Avi, and Virtudes Alba-Fernández. Control estricto de matrices de confusión por medio de distribuciones multinomiales. *GeoFocus Revista Internacional de Ciencia y Tecnología de la Información Geográfica*, pages 215–226, 07 2018.
- [20] Xiaole Zhong and J. Jean Chen. Variations in the frequency and amplitude of resting-state eeg and fmri signals in normal adults: The effects of age and sex. *Computational Intelligence and Neuroscience*, page 2020.10.02.323840, Jenuary 2020. Frequency and amplitude features of both resting-state electroencephalography (EEG) and functional magnetic resonance imaging (fMRI) are crucial metrics that reveal patterns of brain health in aging.
- [21] Aihua Zhang, Bin Yang, and Ling Huang. Feature extraction of eeg signals using power spectral entropy. In *2008 International Conference on BioMedical Engineering and Informatics*, volume 2, pages 435–439, 2008.
- [22] Takashi Kuremoto, Masanao Obayashi, Shingo Mabu, and Kunikazu Kobayashi. *Mental Task Recognition by EEG Signals: A Novel Approach with ROC Analysis*, pages 1,*. IntechOpen, 07 2018.